PENCARIAN ALAMAT FASILITAS UMUM MENGGUNAKAN *METODE VECTOR SPACE MODEL* (STUDI KASUS KOTA PEKANBARU)

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

DEVI BASUMA

10651004331

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU

PEKANBARU

2013

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada saat ini kota Pekanbaru mengalami perkembangan yang sangat pesat dalam pembangunan. Ada banyak fasilitas umum di kota Pekanbaru yang tersebar. Namun fasilitas tersebut saat ini belum dapat dicari dengan mudah. Melihat keadaan tersebut penulis ingin memberikan dokumentasi fasilitas umum yang lengkap. Sehingga masyarakat dengan mudah mendapatkan lokasi yang dicari.

Kebanyakan aplikasi yang memberikan fasilitas pencarian pada saat ini hanya menerapkan *query "%LIKE%"*, salah satunya tugas akhir yang diteliti oleh Alumni Angkatan 2005, jurusan Sistem Informasi UIN Suska Riau dengan judul "Sistem Informasi Geografis Madrasah Aliyah Kabupaten Kampar. Hal ini menyebabkan sejumlah penelitian terhadap mesin pencari (*search engine)* menyimpulkan bahwa rata-rata kesalahan pengerjaan kata kunci yang dilakukan penguna cukup tinggi (Benisius, 2011) sehingga pencarian tidak mendukung *partial matching* dan hasil yang ditemukan tidak maksimal.

Salah satu cara untuk mengatasi masalah tersebut adalah dengan menerapkan aplikasi mesin pencari menggunakan metode *Vektor Space Model*. *Vektor Space Model* (VSM) merupakan salah satu metode yang dapat mengurutkan hasil pencarian sesuai dengan kemiripan antara data yang ada pada *database* dengan kata kunci yang diberikan. Data dengan tingkat kemiripan lebih tinggi memiliki bobot yang lebih tinggi dan akan berada pada urutan pertama sehingga hasil pencarian akan menjadi lebih akurat.

Pada penelitian sebelumnya yang berkaitan dengan pencarian kata menggunakan *Vector Space Model* dibuat oleh Heru Andi Darmawan, dkk (2012) tentang "Rancang Bangun Aplikasi *Search Engine* Tafsir Al-Quran Menggunakan Teknik *Text Mining* Dengan Algoritma *Vector Space Model"*. Pada aplikasi tersebut penulis memberikan fasilitas pencarian yang dapat mempermudah pencarian informasi mengenai kandungan ayat-ayat suci al-quran.

Pencarian lokasi menggunakan *google maps* pada lokasi yang ada di kota Pekanbaru tidak semuanya tercantum dalam hasil pencarian. Oleh karena itu, penulis ingin menambahkan alamat-alamat fasilitas umum yang tidak tercantum didalam *database google maps.* Diharapkan dengan pendataan yang lebih lengkap terhadap falitas umum dapat memudahkan masyarakat dalam menemukan lokasi. Fasilitas umum yang akan penulis data diantaranya fasilitas Rumah Sakit, Hotel, Mesjid, dan SPBU.

Adapun penelitian yang akan saya bangun adalah Aplikasi pencarian alamat fasilitas umum yang ada di kota Pekanbaru. Pada tugas akhir ini saya menggunakan metode *Vector Space Model* untuk melakukan pencarian alamat. Serta membangun aplikasi ini dengan judul "Pencarian Alamat Fasilitas Umum Menggunakan Metode *Vector Space Model* (Studi Kasus Kota Pekanbaru)" diharapkan masyarakat dapat dengan mudah mencari alamat fasilitas umum di kota Pekanbaru.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan diatas, dapat di rumuskan masalah yaitu : "Bagaimana mencari alamat dengan mudah dengan menerapkan metode *Vector Space Model* kedalam sistem informasi".

1.3 Batasan Masalah

Dengan maksud agar pembahasan dan penyusunan sistem dapat dilakukan secara terarah dan bisa berjalan sesuai dengan yang diharapkan, maka perlu ditetapkan batasan-batasan dari masalah yang dihadapi. Batasan-batasan masalah pada perancangan sistem ini adalah sebagai berikut :

1. Informasi mengenai alamat yang disediakan berupa informasi gambar lokasi, peta/denah lokasi nama jalan yang nantinya diintegrasikan kedalam *Google Maps.*
2. Fasilitas umum hanya terbatas pada alamat Rumah Sakit, Hotel, Mesjid dan SPBU yang ada di Kota Pekanbaru.

1.4 Tujuan

Adapun tujuan yang ingin dicapai yaitu :

1. Menganalisa kata pencarian untuk dapat menemukan alamat lokasi fasilitas umum.
2. Menerapkan *Metode Vector Space Model* untuk memperoleh hasil perangkingan dokumen.
3. Membuat sistem informasi yang dapat dipergunakan dengan mudah oleh masyarakat dalam menemukan alamat fasilitas umum.

1.5 Sistematika Pembahasan

Adapun sistematika dalam penulisan skripsi ini adalah sebagai berikut :

Bab I. Pendahuluan

Bab ini menjelaskan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan dan sistematika penulisan laporan tugas akhir.

Bab II. Landasan Teori

Bab ini menjelaskan tentang teori – teori yang digunakan dan relevan dengan topik tugas akhir dimulai dengan teori dasar mengenai *vector space model*, kemudian teknologi *web* sebagai basis dari sistem yang di buat.

Bab III. Metodologi Penelitian

Pada bab ini dijelaskan mengenai tahapan dalam pelaksanaan penelitian tugas akhir. Tahapan penelitian tugas akhir dimulai dari identifikasi permasalahan hingga diperoleh kesimpulan dari penelitian dan saran yang dapat dipergunakan oleh pihak perusahaan maupun oleh peneliti-peneliti selanjutnya.

Bab IV. Analisa Dan Perancangan Sistem

Bab ini menjelaskan proses analisa dan perancangan sistem yang berkaitan dengan masalah yang diteliti dengan menggunakan *Unified Modeling Language* (UML).

Bab V. Implementasi

Bab ini membahas tentang implementasi secara detail serta memberikan hasil pengujian yang dilakukan secara menyeluruh dan terpadu terhadap rancangan sistem yang dibuat disertai dengan penjelasan untuk kondisi uji yang telah disiapkan.

Bab VI. Penutup

Bab ini berisi suatu kesimpulan terhadap aplikasi program yang sudah dibuat secara keseluruhan, dan dikemukakan saran-saran untuk perbaikan serta pengembangan sistem.

BAB II
LANDASAN TEORI

2.1 Pengertian *Website*

Website adalah sebuah penyebaran informasi melalui internet. Sebenarnya antara www *(world wide web)* dan *web* adalah sama karena kebanyakan orang menyingkat *www* menjadi *web* saja. *Web* merupakan hal yang tidak dapat dipisahkan dari dunia internet. Melalui *web*, setiap pemakai internet bisa mengakses informasi-informasi di situs web yang tidak hanya berupa teks, tetapi juga dapat berupa gambar, suara, film, animasi, dll. Sebenarnya, *web* merupakan kumpulan-kumpulan dokumen yang banyak tersebar di beberapa komputer *server* yang berada di seluruh penjuru dunia dan trehubung menjadi satu jaringan melalui jaringan yang disebut internet.

2.1.1 Jenis *Website*

Website atau situs *web* di bedakan menjadi dua tipe yang berbeda, dibedakan berdasarkan fungsional dan cara kerja *website* tersebut, diklarifikasikan sebagai *web* statis dan *web* dinamis.

1. *Web* statis

   Web statsis merupakan *web* dengan Halaman statis yang isi dan tata letak dengan setiap permintaan tidak dapat dirubah oleh masyarakat umum kecuali seorang *programmer* (manusia / *master web*) secara *manual update* halaman. Halaman HTML sederhana adalah contoh dari konten statis.

2. *Web* dinamis atau *Dynamic web*

   Web dinamis adalah *website* dengan halaman beradaptasi konten mereka dan / atau tampilan tergantung pada *input end-user* / interaksi atau perubahan dalam lingkungan komputasi (pengguna, waktu, *database* modifikasi, dll). Konten dapat diubah pada sisi *client* (komputer pengguna akhir) dengan menggunakan bahasa *script* sisi klien (*JavaScript, JScript, Actionscript*, dll) untuk mengubah elemen DOM (DHTML). Konten dinamis sering dikompilasi pada server menggunakan bahasa server-side scripting (*Perl, PHP, ASP, JSP, ColdFusion*, dll). Kedua pendekatan ini biasanya digunakan dalam aplikasi yang kompleks.

2.1.2 Google Maps

2.1.2.1 Pengertian *Google Maps*

Google maps adalah teknologi dari *google* yang memungkinkan kita melihat peta atau mencari lokasi tertentu secara digital. Teknologi *google* brsifat gratis atau *free*, sifatnya gratis ini memasukkan faktor jalan searah atau dua arah untuk menunjukkan jalan dari lokasi awal ke lokasi tujuan. Penggunaannya yang mudah, bisa memasukkan alamat lokasi dari *longitude* dan *latitudenya*, bisa memasukkan nama dan deskripsi lokasinya juga.

Gambar 2.1 Peta *google* beserta deskripsi lokasi

2.1.2.2 Manfaat *Google Map*

Adanya fasilitas tambahan selain sebagai mesin pencari pada *google maps* ini, tentu sangat membantu para pengguna *internet.* Hal ini khususnya dalam proses pencarian sebuah lokasi yang masih asing. Karena dengan adanya *google maps* ini akan memberikan beberapa manfaat di antaranya adalah :

1. Mempercepat pencarian sebuah lokasi dalam waktu yang singkat.
   Karena dengan tekhnologi *digital sistem* pencarian akan berlangsung dengan cepat.
2. Membantu seseorang yang sedang bepergian untuk mencari jalan yang cepat pada lokasi yang hendak ditujunya.
3. Mempermudah sistem penyimpanan peta.

Karena , dengan teknologi digital menjadikan kita tidak perlu ruang yang khusus untuk menyimpan sebuah peta. Sebab, dalam *google maps* ini data disimpan dalam bentuk digital.

4. Bisa diakses dari mana saja.
   Dengan teknologi *internet*, menjadikan kita bisa mengakses peta tersebut dari berbagai tempat yang memiliki fasilitas *internet*.
5. Mengetahui tempat tempat baru yang mungkin belum kita ketahui sebelumnya.
6. Adanya pembaharuan data yang lebih cepat dari pada menggunakan peta konvensional.

2.2 Fasilitas Umum

2.2.1 Pengertian Fasilitas Umum

Arti definisi/pengertian fasilitas umum adalah fasilitas yang diadakan untuk kepentingan umum. Contoh dari fasilitas umum (fasum) adalah seperti jalan, angkutan umum, saluran air, jembatan, *fly over*, *under pass*, halte, alat penerangan umum, jaringan listrik, banjir kanal, trotoar, jalur *busway*, tempat pembuangan sampah, dan lain sebagainya.

2.3 Information Retrieval

2.3.1 Definisi *Information Retrieval*

ISO 2382/1 mendefinisikan *Information Retrieval* sebagai tindakan, metode dan prosedur untuk menemukan kembali data yang tersimpan, kemudian menyediakan informasi mengenai subyek yang dibutuhkan. Tindakan tersebut *mencakup text indexing*, *inquiry analysis*, dan *relevance analysis*. Data mencakup teks, tabel, gambar, ucapan, dan video. Informasi termasuk pengetahuan terkait yang dibutuhkan untuk mendukung penyelesaian masalah dan akuisisi pengetahuan.

Tujuan dari sistem *Information Retrieval* adalah memenuhi kebutuhan informasi pengguna dengan me-*retrieve* semua dokumen yang mungkin *relevan*, pada waktu yang sama me-*retrieve* sesedikit mungkin dokumen yang tak-*relevan*.

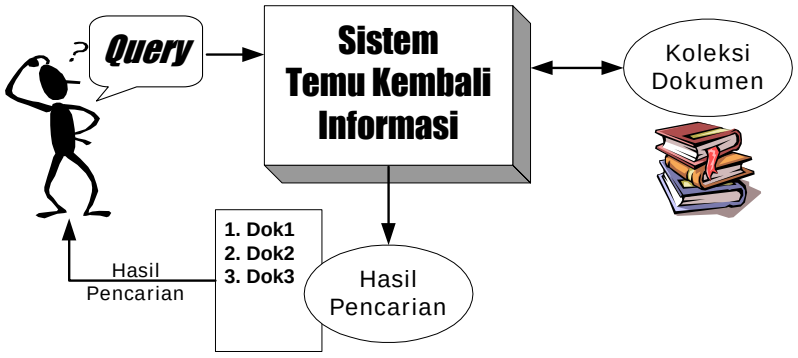
Sistem ini menggunakan fungsi heuristik untuk mendapatkan dokumen-dokumen yang *relevan* dengan *query* pengguna. Sistem *Information Retrieval* yang baik memungkinkan pengguna menentukan secara cepat dan akurat apakah isi dari dokumen yang diterima memenuhi kebutuhannya. Agar representasi dokumen lebih baik, dokumen-dokumen dengan topik atau isi yang mirip dikelompokkan bersama-sama.

2.3.2 Arsitektur Sistem *Information Retrieval*

Secara garis besar arsitektur sistem *Information Retrieval* ada dua pekerjaan yang ditangani oleh sistem ini, yaitu melakukan *pre-processing* terhadap *database* dan kemudian menerapkan metode tertentu untuk menghitung kedekatan (*relevansi* atau *similarity*) antara dokumen di dalam *database* yang telah di-*preprocess* dengan *query* pengguna. Pada tahapan *preprocessing*, sistem yang berurusan dengan dokumen *semi-structured* biasanya memberikan *tag* tertentu pada *term-term* atau bagian dari dokumen; sedangkan pada dokumen tidak terstruktur proses ini dilewati dan membiarkan *term* tanpa imbuhan *tag*. Query yang dimasukkan pengguna dikonversi sesuai aturan tertentu untuk mengekstrak *term-term* penting yang sejalan dengan *term-term* yang sebelumnya telah diekstrak dari dokumen dan menghitung relevansi antara *query* dan dokumen berdasarkan pada *term-term* tersebut. Sebagai hasilnya, sistem mengembalikan suatu daftar dokumen terurut *descending* (ranking) sesuai nilai kemiripannya dengan *query* pengguna.

Setiap dokumen (termasuk *query*) direpresentasikan menggunakan model *bag-of-words* yang mengabaikan urutan dari kata-kata di dalam dokumen, struktur sintaktis dari dokumen dan kalimat. Dokumen ditransformasi ke dalam suatu “tas“ berisi kata-kata independen. *Term* disimpan dalam suatu *database* pencarian khusus yang ditata sebagai sebuah *inverted index*. *Index* ini merupakan konversi dari dokumen asli yang mengandung sekumpulan kata ke dalam daftar kata yang berasosiasi dengan dokumenterkait dimana kata-kata tersebut muncul.

Menurut Mandala dan Setiawan (2002), arsitektur sistem *information retrieval* dapat digambarkan seperti di bawah ini:



Gambar 2.2 Ilustrasi Sistem Temu Kembali Informasi
(Mandala dan Setiawan, 2002)

Ada dua pekerjaan yang ditangani oleh sistem ini, yaitu melakukan *pre-processing* terhadap *database* dan kemudian menerapkan model tertentu untuk menghitung kedekatan (relevansi atau *similarity*) antara dokumen di dalam *database* yang telah dipreproses dengan *query* pengguna.

2.3.2.1 Koleksi Dokumen (*Corpus*)

Istilah *corpus* pada prinsipnya bermakna koleksi dokumen yang diindeks dan dijadikan target pencarian. Suatu *corpus* modern memiliki beberapa karakteristik yakni (McEnery & Wilson, 2001) :

1. *Sampling & representativeness*
2. *Finite size*
3. *Machine-readable form*
4. *A standard reference*

Suatu *corpus* pengujian sistem *information retrieval* terdiri dari:

1. Koleksi dokumen.
2. Topik-topik, yang dapat digunakan sebagai *query.*
3. *Relevance judgement*, sebagai daftar dokumen yang relevan dengan topik-topik yang tersedia.

2.3.2.2 *Text Preprocessing*

Pada tahapan *preprocessing*, *query* yang dimasukkan pengguna dikonversi sesuai aturan tertentu untuk mengekstrak *term-term* penting yang sejalan dengan *term-term* yang sebelumnya telah diekstrak dari dokumen dan menghitung relevansi antara *query* dan dokumen berdasarkan pada *term-term* tersebut. Sebagai hasilnya, sistem mengembalikan suatu daftar dokumen terurut descending (ranking) sesuai nilai kemiripannya dengan *query* pengguna (Krzysztof J.Cios, 2007).

Adapun tahap *Text Preprocessing* terdiri dari :

1. Pembangunan *Index*

Pembangunan *index* dari koleksi dokumen merupakan tugas pokok pada tahapan *preprocessing* di dalam sistem information retrieval. Kualitas *index* mempengaruhi efektivitas dan efisiensi sistem information retrieval. *Index* dokumen adalah himpunan *term* yang menunjukkan isi atau topik yang dikandung oleh dokumen.

Index akan membedakan suatu dokumen dari dokumen lain yang berada di dalam koleksi. Ukuran *index* yang kecil dapat memberikan hasil buruk dan mungkin beberapa *item* yang relevan terabaikan. *Index* yang besar memungkinkan ditemukan banyak dokumen yang relevan tetapi sekaligus dapat menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pencarian (Trunojoyo, 2010).

Terdapat lima langkah pembangunan *inverted index* (Trunojoyo, 2010), yaitu:

1. Penghapusan *format* dan *markup* dari dalam dokumen

   Tahap ini menghapus semua *tag markup* dan *forma*t khusus dari dokumen, terutama pada dokumen yang mempunyai banyak *tag* dan format seperti dokumen (X)HTML.

2. Pemisahan rangkaian kata (*tokenization*)

   Tokenization adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi token atau potongan kata tunggal atau *termmed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu

seperti tanda baca dan mengubah semua token ke bentuk huruf kecil (*lower case*).

3. Penyaringan (*filtration*)

   Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. *Term* yang sering digunakan dianggap sebagai *stop-word* dan dihapus.

   Penghapusan *stop-word* dari dalam suatu koleksi dokumen pada satu waktu membutuhkan banyak waktu. Solusinya adalah dengan menyusun suatu pustaka *stop-word* atau *stop-list* dari *term* yang akan dihapus (Manning, 2008).

4. Konversi *term* ke bentuk akar (*stemming*)

   Stemming adalah salah satu cara yang digunakan untuk meningkatkan performa sistem *information retrieval* dengan cara mentransformasi kata-kata dalam sebuah dokumen teks ke bentuk kata dasarnya, contohnya kata-kata menyukseskan, tersukseskan dan disukseskan akan ditransformasi ke *stem* yang sama yaitu sukses. Algoritma *stemming* untuk bahasa yang satu berbeda dengan algoritma *stemming* untuk bahasa lainnya. Sebagai contoh bahasa Inggris memiliki morfologi yang berbeda dengan bahasa Indonesia sehingga algoritma *stemming* untuk kedua bahasa tersebut juga berbeda.

   Tidak banyak algoritma yang dikhususkan untuk *stemming* bahasa Indonesia dengan berbagai keterbatasan didalamnya, diantaranya :

a. Algoritma Porter, Algoritma ini membutuhkan waktu yang lebih singkat dibandingkan dengan *stemming* menggunakan Algoritma Nazief & Adriani, namun proses *stemming* menggunakan Algoritma Porter memiliki presentase keakuratan (presisi) lebih kecil dibandingkan dengan *stemming* menggunakan Algoritma Nazief & Adriani.
b. Algoritma Nazief & Adriani, algoritma *stemming* untuk teks berbahasa Indonesia yang memiliki kemampuan presentase keakuratan (presisi) lebih baik dari algoritma lainnya. Algoritma ini sangat dibutuhkan dan menentukan dalam proses sistem *information retrieval* dalam dokumen Indonesia (Agusta, 2009). Algoritma Nazief & Adriani mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan. Pengelompokan ini termasuk imbuhan di depan (awalan), imbuhan kata di belakang (akhiran), imbuhan kata di tengah (sisipan) dan kombinasi imbuhan pada awal dan akhir kata (konfiks).

   Algoritma ini menggunakan kamus kata keterangan yang digunakan untuk mengetahui bahwa proses stemming telah mendapatkan kata dasar (Nazief, B.A.A. dan Andriani, M. 1996).

   DP + DP + DP + root word + DS + PP + P

   Langkah-langkah pada Algoritma Nazief & Adriani adalah:

   1. Kata yang belum di-*stemming* dicari pada kamus. Jika kata itu langsung ditemukan, berarti kata tersebut adalah kata dasar. Kata tersebut dikembalikan dan algoritma dihentikan.

2. Hilangkan *inflectional suffixes* terlebih dahulu. Jika hal ini berhasil dan *suffix* adalah partikel (“lah” atau ”kah”), langkah ini dilakukan lagi untuk menghilangkan *inflectional possessive pronoun suffixes* (“ku”, “mu” atau ”nya”).
3. *Derivational suffix* kemudian dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational suffix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
4. Kemudian *derivational prefix* dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational prefix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
5. Setelah tidak ada lagi imbuhan yang tersisa, maka algoritma ini dihentikan kemudian kata dasar tersebut dicari pada kamus, jika kata dasar tersebut ketemu berarti algoritma ini berhasil tapi jika kata dasar tersebut tidak ketemu pada kamus, maka dilakukan *recoding*.
6. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus juga maka algoritma ini mengembalikan kata yang asli sebelum dilakukan stemming.

Kelebihan pada algoritma Nazief dan Andriani ini adalah bahwa algoritma ini memperhatikan kemungkinan adanya partikel-partikel yang mungkin mengikuti suatu kata berimbuhan. Sehingga kita dapat melihat pada rumus untuk algoritma ini yaitu adanya penempatan *possesive pronoun* dan juga partikel yang mungkin ada pada suatu kata berimbuhan. Akhir dari algoritma ini yaitu apabila pemotongan semua imbuhan telah berhasil dan hasil pemotongan imbuhan tersebut terdapat pada kamus maka algoritma ini dapat dikatan berhasil dalam penentuan kata dasarnya. Dan apabila sebaliknya bahwa algoritma ini setelah dilakukan pemotongan kata dan tidak terdapat pada kamus maka kata berimbuhan yang telah mengalami pemotongan dikembalikan ke keadaan semula.

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut : Cari kata yang akan distemming dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*, maka algoritma berhenti.*Inflection suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa partikel (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *possesive pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.

1. Hapus *Derivation suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
2. Hapus *Derivation prefix*. Jika pada langkah 3 ada *sufiks* yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5.
3. Jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama maka algoritma berhenti. Melakukan *recoding*.
4. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

   Tipe awalan ditentukan melalui langkah-langkah berikut:

   1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
   2. Pemberian bobot terhadap *term* (*weighting*)

      Setiap *term* diberikan bobot sesuai dengan skema pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya.

2 .3.2.3 Pembobotan Kata

Salah satu cara untuk memberi bobot terhadap suatu kata adalah memberikan nilai jumlah kemunculan suatu kata (*term frequency*) sebagai bobot. Semakin besar kemunculan suatu kata dalam dokumen akan memberikan nilai kesesuian yang semakin besar. Faktor lain yang diperhatikan dalam pemberian bobot adalah kejarangmunculan kata (*term scarcity*) dalam koleksi. Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon terms*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*) (Mandala dan Setiawan, 2002).

2.4 Membandingkan Kata Kunci dan Koleksi Dokumen

Sistem information retrieval menerima *query* dari pengguna, kemudian melakukan perangkingan terhadap dokumen pada koleksi berdasarkan kesesuaiannya dengan *query*.

Hasil perangkingan yang diberikan kepada pengguna merupakan dokumen yang menurut sistem relevan dengan *query*. Model sistem *information retrieval* menentukan *detail* sistem *information retrieva*l yaitu meliputi representasi dokumen maupun *query*, fungsi pencarian (*retrieval function*) dan notasi kesesuaian (*relevance notation*) dokumen terhadap *query*.

2.4.1 Model dalam Sistem *Information Retrieval*

Di dalam bidang sistem *information retrieval*, dikenal berbagai model untuk menilai secara objektif presisi dari suatu pencarian, antara lain model Boolean (*Boolean Model*), model ruang vektor (*Vector Space Model*) dan model probabilistik (*Probabilistic Model*).

2.4.2 Model Boolean

Model Boolean adalah model yang paling awal dikenal dan paling mudah untuk diimplementasikan. Hanya saja, model Boolean tidak dapat memberikan hasil yang diharapkan dan sangat lambat dalam run-time (Jaya, 2007).

Model Boolean dalam *information retrieval*merupakan model yang paling sederhana. Model ini berdasarkan teori himpunan dan aljabar Boolean. Dokumen adalah himpunan dari istilah (*term*) dan *query* adalah pernyataan Boolean yang ditulis pada *term*. Dokumen diprediksi apakah relevan atau tidak. Model ini menggunakan operator boolean. Istilah (*term*) dalam sebuah *query* dihubungkan dengan menggunakan operator *AND, OR* atau *NOT*.

Beberapa karakteristik dari model boolean dalam sistem *information retrieval adalah* :

1. Model Boolean merupakan model sederhana yang menggunakan teori dasar himpunan sehingga mudah diimplementasikan.
2. Model Boolean tidak menggunakan peringkat dokumen yang terambil. Dokumen yang terambil hanya dokumen yang benar-benar sesuai dengan pernyataan boolean / *query* yang diberikan Sehingga dokumen yang terambil bisa sangat banyak atau bisa sedikit. Akibatnya ada kesulitan dalam mengambil keputusan.
3. Teori himpunan memang mudah, namun tidak demikian halnya dengan pernyataan Boolean yang bisa kompleks. Akibatnya pengguna harus memiliki pengetahuan banyak mengenai *query* dengan boolean agar pencarian menjadi efisien.

2.4.3 Model Probabilistik

Sistem information retrieval juga memperkenalkan model probabilistik. Model ini mengurutkan dokumen dalam urutan menurun terhadap peluang relevansi sebuah dokumen pada informasi yang dibutuhkan (Ramadhany, 2008).

2.4.4 Model Ruang Vektor

Model berikutnya adalah model ruang vektor yang saat ini sangat populer dalam sistem *information retrieval*. Model ini berhasil memberikan hasil yang lebih baik dibandingkan model Boolean. Model ini juga dapat menampilkan hasil temu balik secara terurut (Jaya, 2007). Model ruang vektor tidak membutuhkan komputasi yang berlebihan sehingga waktu untuk mengeksekusi akan semakin cepat dan lebih efektif (Ramadhany, 2008). Beberapa karakteristik dari model ruang vektor dalam *information retrieval* adalah :

1. Model vektor berdasarkan *keyterm*
2. Model vektor mendukung *partial matching* (sebagian sesuai) dan penentuan peringkat dokumen
3. Prinsip dasar model vektor adalah sebagai berikut :
   a) Dokumen direpresentasikan dengan menggunkan vektor key*term*
   b) Ruang dimensi ditentukan oleh *keyterms*
   c) *Query* direpresentasikan dengan menggunakan vektor *keyterm*
   d) Kesamaan *document keyterm* dihitung berdasarkan jarak vektor
4. Model ruang vektor memerlukan
   a) Bobot *keyterm* untuk vektor dokumen
   b) Bobot *keyterm* untuk *query*
   c) Perhitungan jarak untuk vektor *document keyterm*
5. Kinerja
   a) Efisien
   b) Mudah dalam representasi
   c) Dapat diimplementasikan pada *document matching*

Prosedur model ruang vektor dapat dikelompokkan menjadi tiga tahap yaitu :

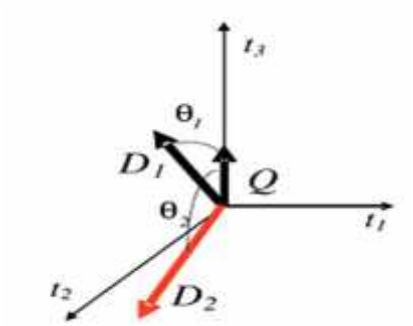
1. Pengindeks-an dokumen
2. Pembobotan indeks, untuk menghasilkan dokumen yang relevan
3. Memberikan peringkat dokumen berdasarkan ukuran kesamaan (*similarity measure*)

Pada model ruang vektor, setiap dokumen di dalam *database* dan *query* pengguna direpresentasikan oleh suatu vektor multi-dimensi (Nicola Polettini, 2004). Prinsip utamanya adalah *query* diubah menjadi vektor *query* dan dokumen-dokumen di dalam koleksi dokumen diubah menjadi vektor-vektor dokumen (Salton, 1975).

2.4.4.1 Rumus Relevansi

Penentuan relevansi dokumen dengan *query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara vektor dokumen dengan vektor *query*. Semakin "sama" suatu vektor dokumen dengan vektor *query* maka dokumen dapat dipandang semakin relevan dengan *query*. Salah satu pengukuran kesesuaian yang baik adalah dengan memperhatikan perbedaan arah (*direction difference*) dari kedua vektor tersebut. Perbedaan arah kedua vektor dalam geometri dapat dianggap sebagai sudut yang terbentuk oleh kedua vektor.

Gambar 2.3 mengilustrasikan kesamaan antara dokumen D_1 dan D_2 dengan *query* Q. Sudut θ_1 menggambarkan kesamaan dokumen D_1 dengan *query* sedangkan sudut θ_2 mengambarkan kesamaan dokumen D_2 dengan *query*.



Gambar 2.3 Representasi Grafis Sudut Vektor Dokumen dan *Query*
(Mandala dan Setiawan, 2002)

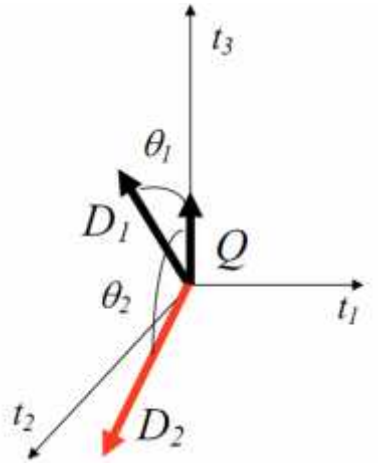
Perhitungan kesamaan antara vektor *query* dan vektor dokumen dilihat dari sudut yang paling kecil. Sudut yang dibentuk oleh dua buah vektor dapat dihitung dengan melakukan perkalian dalam (*inner product*), sehingga rumus relevansinya adalah:

$$R(Q,D) = \cos\theta = \frac{Q \bullet D}{|Q||D|} \quad \text{.....(2.1)}$$

Jika *Q* adalah vektor *query* dan *D* adalah vektor dokumen, yang merupakan dua buah vektor dalam ruang berdimensi-*n*, dan θ adalah sudut yang dibentuk oleh kedua vektor tersebut. Maka

$$Q \bullet D = |Q||D| \cos\theta \quad \text{.....(2.2)}$$

Dapat dilihat pada Gambar 2.4 berikut :



Gambar 2.4 Besar sudut antara vektor *query* dan vektor dokumen

dimana $Q \bullet D$ adalah hasil perkalian dalam (*inner product*) kedua vektor, sedangkan

$$|Q| = \sqrt{\sum_{i=1}^{n}(wq_i)^2} \text{ dan } |D| = \sqrt{\sum_{i=1}^{n}(wd_{ij})^2} \quad \text{.....(2.3)}$$

merupakan panjang vektor atau jarak *Euclidean* suatu vektor dengan titik nol.

Metode pengukuran kesesuaian ini memiliki beberapa keuntungan, yaitu adanya normalisasi terhadap panjang dokumen. Hal ini memperkecil pengaruh panjang dokumen. Jarak *Euclidean* (panjang) kedua vektor digunakan sebagai faktor normalisasi. Hal ini diperlukan karena dokumen yang panjang cenderung mendapatkan nilai yang besar dibandingkan dengan dokumen yang lebih pendek. Dengan demikian, ukuran kosinus sudut antara kedua vektor dapat dinyatakan sebagai berikut (Mandala, 2006) :

$$R\ Q,D\ = \cos\theta = \frac{Q \bullet D}{|Q||D|} = \frac{\sum_{i=1}^{n} (wq_i . wd_{ij})}{\sum_{i=1}^{n} wqi^{2} \cdot \sum_{i=1}^{n} (wdj)^{2}} \text{ ……...(2.4)}$$

dimana :

wq_i = bobot pada *query* ke-i = tf x idf …….....(2.5)

wd_{ij} = bobot pada dokumen ke-i istilah ke-j = tf x idf ……...(2.6)

tf = *term frequency* = frekuensi kemunculan istilah pada dokumen atau *query*

Idf dapat ditentukan dengan menggunakan rumus :

$$Idf = \log\left[\frac{N}{df}\right] \text{ …….....(2.7)}$$

Dimana :

Idf = *inverse document frequency* (pembobotan secara global)

N = total dokumen

df = jumlah dukumen yang mengandung istilah tertentu

2.4.4.2 Contoh penggunaan *vector space model*

Contoh implementasi sederhana dari TF-IDF adalah sebagai berikut : Kata kunci (kk) = pengetahuan logistik.

Dokumen 1 (D1) = manajemen transaksi logistik

Dokumen 2 (D2) = pengetahuan antar individu

Dokumen 3 (D3) = dalam manajemen pengetahuan terdapat *transfer* pengetahuan logistik.

<u>Tahapan - tahapan yang dilakukan :</u>

1. Menghilangan tanda baca.

   Kata kunci = pengetahuan logistik.

2. Mengubah istilah ke bentuk huruf kecil.

   Kata kunci = pengetahuan logistik.

3. Menerapkan *stopword removal.*

   Adapun daftar *stop word* dari tiga contoh dokumen diatas adalah : yang, dalam, dapat, untuk, tersebut, dengan, adalah, dan, bagi, besar, secara.

   Kata kunci = pengetahuan logistik.

4. Menerapkan *stemming* (mengembalikan kata ke kata dasar).

Adapun daftar *stemming* dari tiga contoh dokumen diatas adalah : milik, hasil, manfaat, selesai, bantu, dukung, putus, kumpul, basis, proses, beri, timbang, ambil, rupa, masalah, mungkin.

Kata kunci = pengetahuan logistik.

5. Pembobotan, setelah semua dokumen *dipreprocessing* tiap *term* dipisah dan dimasukkan ke dalam tabel *indexing.*

Setelah proses tersebut, maka pada algoritma TF/IDF digunakan rumus untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci dengan rumus yaitu :

$W_{dt} = tf_{dt} * IDF_t$

Dimana:

d = dokumen ke-d

t = kata ke-t dari kata kunci

W = bobot dokumen ke-d terhadap kata ke-t

tf = banyaknya kata yang dicari pada sebuah dokumen

IDF = *Inversed Document Frequency*

$IDF = log_2(\frac{D}{df})$

D = total dokumen

df = banyak dokumen yang mengandung kata yang dicari

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses sorting/pengurutan dimana semakin besar nilai W, semakin besar tingkat similaritas dokumen tersebut terhadap kata kunci, demikian sebaliknya.

Setelah dilakukan tahap tokenizing dan prosesfiltering, maka kata antar pada dokumen 2 serta kata dalam dan terdapat pada dokumen 3 dihapus. Berikut ini adalah tabel 2.1 perhitungan TF/IDF.

Tabel 2.1 Contoh perhitungan TF/IDF

	tf				df	D/df	*IDF = log (D/df)*	W			
Token	**kk**	**D1**	**D2**	**D3**				**kk**	**D1**	**D2**	**D3**
manajemen	0	1	0	1	2	1.5	0.176	0		0	0.176
transaksi	0	1	0	0	1	3	0.477	0		0	0
logistik	1	1	0	1	2	1.5	0.176	0.176	0.176	0	0.176
transfer	0	0	0	1	1	3	0.477	0		0	0.477
pengetahuan	1	0	1	2	2	1.5	0.176	0.176	0	0.176	0.352
individu	0	0	1	0	1	3	0.477	0	0	0.477	0
	total							**0.352**	**0.829**	**0.653**	**1.181**

bobot (W) untuk D1 = 0.176 + 0 = 0.176

bobot (W) untuk D2 = 0 + 0.176 = 0.176

bobot (W) untuk D3 = 0.176 + 0.352 = 0.528

Dari contoh studi kasus di atas, dapat diketahui bahwa nilai bobot (W) dari D1 dan D2 adalah sama. Apabila hasil pengurutan bobot dokumen tidak dapat mengurutkan secara tepat, karena nilai W keduanya sama, maka diperlukan proses perhitungan dengan algoritma *vector space model*. Ide dari metode ini adalah dengan menghitung nilai *cosinus* sudut dari dua vektor, yaitu W dari tiap dokumen dan W dari kata kunci.

Vector space model adalah suatu model yang digunakan untuk mengukur kemiripan antara suatu dokumen dengan suatu *query.* Pada model ini, *query* dan dokumen dianggap sebagai vektor-vektor pada ruang n-dimensi, dimana n adalah jumlah dari seluruh *term* yang ada dalam leksikon. *Leksikon* adalah daftar semua term yang ada dalam indeks. Salah satu cara untuk mengatasi hal tersebut dalam *model vector space* adalah dengan cara melakukan perluasan vektor. Proses perluasan dapat dilakukan pada vektor *query*, vektor dokumen, atau pada kedua vektor tersebut.

Pada algoritma *vector space model* gunakan rumus untuk mencari nilai *cosinus* sudut antara dua vector dari setiap bobot dokumen (WD) dan bobot dari kata kunci (WK). Rumus yang digunakan adalah sebagai berikut :

$$Cosine(d_j, q) = \frac{d_j \bullet q}{d_j \, |q|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sum_{i=1}^{t} w_{ij}^{2} \cdot \sum_{i=1}^{n} (w_{iq})^{2}}$$

Apabila studi kasus pada algoritma TF/IDF di atas dicari nilai *cosinus* sudut antara vector masing-masing dokumen dengan kata kunci, maka hasil yang didapatkan akan lebih presisi. Seperti yang ditunjukan tabel 2.2

Tabel 2.2 Contoh perhitungan TF/IDF

Token	kk	D1	D2	D3	kk*D1	kk*D2	kk*D3
manajemen	0	0.031	0	0.031	0	0	0
transaksi	0	0.028	0	0	0	0	0
logistik	0.031	0.031	0	0.031	0	0.031	0.031
transfer	0	0	0	0.028	0	0	0
pengetahuan	0.031	0	0.031	0.124	0	0.031	0.062
individu	0	0	0.228	0	0	0	0
	sqrt(kk)	sqrt(Di)			sqrt(kk*Di)		
	0.249	0.539	0.509	0.643	0.031	0.031	0.093

Selanjutnya menghitung nilai cosinus sudut antara vector kata kunci dengan tiap dokumen dengan menngunakan rumus:

$$Cosine(D_i) = sum\ kk\ D_i\ /sqrt\ (kk) \square sqrt(D_i)$$

1. Untuk Dokumen 1 (D1)

   Cosine (D1) = sum (kk . D1) / (sqrt(kk) * sqrt(D1))

   = 0.031 / (0.249 * 0.539)

   = 0.231

2. Untuk Dokumen 2 (D2)

   Cosine (D2) = sum (kk . D2) / (sqrt(kk) * sqrt(D2))

   = 0.031 / (0.249 * 0.509)

   = 0.245

3. Untuk Dokumen 2 (D3)

   Cosine (D3) = sum (kk . D3) / (sqrt(kk) * sqrt(D3))

   = 0.093 / (0.249 * 0.643)

= 0.581

Sesuai perhitungan diatas maka nilai cosinus setiap dokumen telah didapat, seperti tabel 2.3.

Tabel 2.3 hasil perhitungan *vector space model*

	D1	D2	D3
Cosine	0.231	0.245	0.581
	Rank 3	Rank 2	Rank 1

Dari hasil akhir tersebut dapat diketahui bahwa dokumen 3 (D3) memiliki tingkat similaritas tertinggi terhadap kata kunci, kemudian disusul dengan D2 dan D1.

2.5 Precision and Recall

Untuk menghitung performansi sistem temu kembali, yaitu *recall* dan *precision*. *Recall* dinyatakan sebagai bagian dari dokumen relevan dalam dokumen yang ditemukan, *recall* merupakan jumlah dokumen yang seharusnya terambil oleh sistem berdasarkan perhitungan manual.

$$recall = \frac{Jumlah\ dokumen\ relevan\ yang\ berhasil\ ditemukan}{jumlah\ seluruh\ dokumen\ relevan} \quad \text{……..……..(2.8)}$$

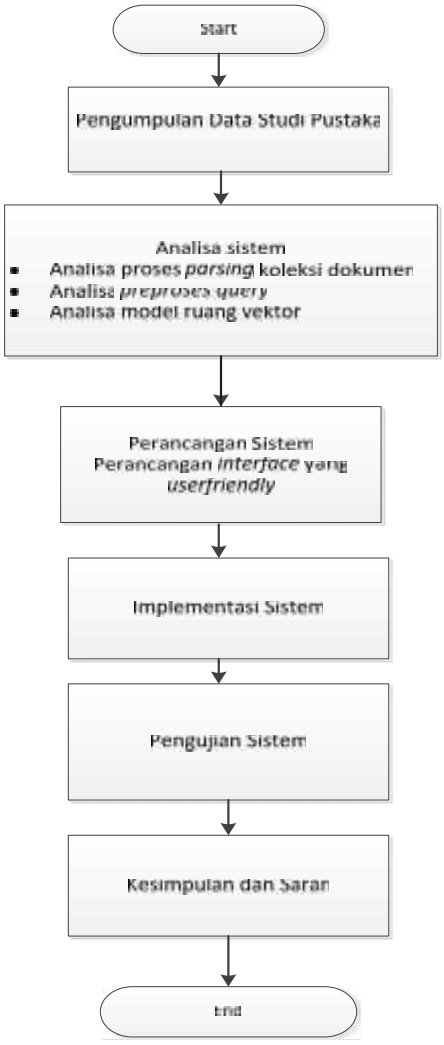
Sedangkan *precision* dinyatakan sebagai bagian dokumen relevan yang ditemukan, jumlah dokumen yang ditemukan pada *precision* ini berdasarkan perhitungan oleh sistem temu balik informasi.

$$precision = \frac{Jumlah\ dokumen\ relevan\ yang\ berhasil\ ditemukan}{jumlah\ seluruh\ dokumen\ yang\ ditemukan} \quad \text{……..(2.9)}$$

Keduanya menggambarkan performansi dari sistem temu balik informasi dengan melakukan perhitungan terhadap jumlah dokumen relevan hasil pencarian. Pengukuran *recall* dan *precision* ini merupakan perhitungan yang dilakukan terhadap kumpulan dokumen hasil pencarian (*set based measure*) secara keseluruhan.

BAB III
METODOLOGI PENELITIAN

Metodologi penelitian merupakan sistematika tahapan yang dilaksanakan selama pembuatan tugas akhir. Adapun tahapan yang dilalui dalam pelaksanaan penelitian ini adalah sebagai berikut:



Gambar 3.1 Bagan Penyusunan Tugas Akhir

3.1 Pengumpulan Data

Metode yang penulis lakukan untuk memperoleh informasi atau pengumpulan data pada penelitian ini yaitu metode studi pustaka. Studi pustaka berfungsi untuk mendukung penelitian yang akan dilaksanakan. Pengumpulan teori-teori yang mendukung dalam penelitian ini merupakan kegiatan dalam studi pustaka. Sumber yang digunakan dapat berupa buku, jurnal dan tulisan penelitian tentang pencarian lokasi, metode vektor dalam pencarian lokasi atau artikel-artikel yang membahas kasus yang sama dengan kasus dalam laporan ini.

3.2 Analisa Sistem

Analisa merupakan metode yang dilakukan setelah pengumpulan data dan informasi mengenai kasus yang diangkat pada penelitian tugas akhir ini. Analisa berarti metode yang khusus untuk menganalisis masalah dibangunnya sistem dan hasil akhir yang ingin dicapai dari pembuatan sistem. Analisa utama yang akan dilakukan adalah :

1. Analisa proses *parsing* koleksi dokumen, dalam proses ini terdapat beberapa tahapan, yaitu :
    a. Mengumpulkan dokumen kedalam sebuah *corpus* (koleksi dokumen).
    b. Melakukan pemisahan rangkaian kata atau *tokenization*.
    c. Melakukan penyaringan kata (*filtration*) berdasarkan daftar *stop list*.
    d. Konversi istilah ke bentuk dasar istilah tersebut (*stemming*).
    e. Melakukan *indexing* dan memberikan bobot terhadap setiap istilah (*weighting*).
2. Analisa *preproses* pada *query* yang diinputkan pengguna, pada analisa ini pada *query* dilakukan *indexing*, sama dengan tahapan *indexing* pada koleksi dokumen, lalu dilakukan pencocokan terhadap koleksi dokumen.
3. Analisa model ruang vektor, prinsip utama dari model ini adalah mengubah *query* yang diinputkan pengguna yang telah di preproses menjadi vektor *query* dan dokumen pada koleksi dokumen (*corpus*) menjadi vektor dokumen. Semakin “sama” suatu vektor dokumen dengan

vektor *query* maka dokumen dapat dipandang semakin relevan dengan *query*.

3.3 Perancangan Sistem

Perancangan berarti metode yang khusus digunakan untuk merancang sistem yang telah dianalisa dengan tujuan untuk memberikan kemudahan dan menyederhanakan suatu proses atau jalannya aliran data, perancangan terhadap model, dan merancang rancang bangun sistem. Adapun rancangan utama sistem yaitu :

1. Perancangan *interface* sistem yang *userfriendly* sehingga memudahkan pengguna dalam mengakses sistem.
2. Perancangan aplikasi pencarian alamat menggunakan metode *vector space model*.

3.4 Implementasi Sistem dan Pengujian

Pada proses implementasi ini akan dilakukan pembuatan modul yang telah dirancang dan dianalisa selanjutnya diimplementasikan pada bahasa pemrograman dan dilakukan pengujian untuk mengetahui tingkat keberhasilan aplikasi yang telah ada. Berikut adalah spesifikasi lingkungan implementasi perangkat keras dan perangkat lunak :

1. Perangkat keras

   Processor : *Intel(R) Core(TM) i3 CPU M 430 @2.27GHz*

   Memori (RAM) : 2.00 GB

2. Perangkat Lunak

   SistemOperasi : *Windows 7 Home Premium*

   BahasaPemrograman : Php

   DBMS : mySQL

   *Tools*Perancangan : *Dreamweaver 8*

3.5 Pengujian Sistem

Pengujian merupakan tahapan dimana sistem akan dijalankan. Tahap pengujian diperlukan sebagai ukuran bahwa sistem dapat dijalankan sesuai dengan tujuan. Pengujian aplikasi pencarian alamat menggunakan metode *vector space model*. ini dilakukan dengan cara mengukur kualitas dari informasi yang dikembalikan sistem berdasarkan *query* yang diinputkan pengguna. Ukuran yang digunakan untuk mengukur kualitas pencarian menggunakan algoritma *vector space model* ini menggunakan *recall* dan *precision*.

3.6 Kesimpulan dan Saran

Tahapan kesimpulan dan saran merupakan akhir dari penelitian tugas akhir. Tahapan kesimpulan membahas hasil evaluasi dari seluruh kegiatan yang dilakukan dalam melakukan penelitian terhadap pembuatan aplikasi pencarian alamat menggunakan metode *vector space model* serta memberikan saran-saran untuk menyempurnakan dan mengembangkan penelitian sistem temu balik informasi selanjutnya.

BAB IV

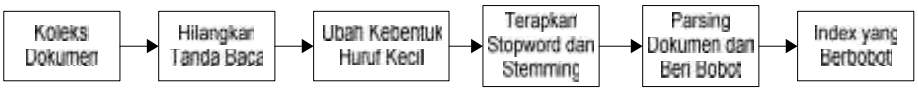
ANALISA DAN PERANCANGAN

Bab analisa dan perancangan ini berisi perincian tahapan dari sistem *Information Retrieval* yang akan dibangun, model yang digunakan adalah model ruang vektor dengan menerapkan *stemming.*

4.1 Analisa Sistem *Information Retrieval* (IR)

Secara garis besar, ada tiga tahapan yang ditangani oleh sistem ini, yaitu melakukan preproses terhadap dokumen, melakukan preproses terhadap *query* pengguna dan menerapkan metode tertentu dalam hal ini menggunakan model ruang vektor untuk menghitung kedekatan (relevansi / *similirity*) antara dokumen dan *query* pengguna tersebut. Adapun tiga tahapan tersebut, yaitu bisa dilihat pada gambar 4.1 berikut :

1. Tahapan Preproses Dokumen



Gambar 4.1 Tahapan Preproses Dokumen

a. Menyimpan dokumen kedalam koleksi dokumen

   Sebelum dilakukan tahapan preproses, semua dokumen yang akan dicari disimpan dalam sebuah koleksi dokumen. Adapun dokumen yang akan dijadikan koleksi dokumen adalah fasilitas umum yang ada dikota Pekanbaru disimpan didalam *dbms* MySQL.

b. Menghilangkan tanda baca pada dokumen

   Semua tanda baca yang ada pada koleksi dokumen akan dihilangkan. Penghilangan tanda baca tersebut tergantung dari koleksi tanda baca yang

ada pada *database*, misalnya : !, ?, “” dan lain sebagainya. Adapun fungsi yang akan digunakan pada tahapan ini, yaitu :

function stop_kar (teks)

Masukan : dokumen dalam bentuk teks yang masih memiliki tanda baca.

Keluaran : tanda baca yang ada pada dokumen dihapus.

c. Mengubah dokumen kebentuk huruf kecil

Tahapan preproses dokumen berikutnya adalah mengubah koleksi dokumen ke bentuk huruf kecil. Adapun fungsi yang akan digunakan pada tahapan ini, yaitu :

function huruf_kecil (teks)

Masukan : dokumen dalam bentuk teks yang belum *lower case*.

Keluaran : seluruh teks yang ada pada dokumen dijadikan *lower case*.

d. Menerapkan *stop word removal*

Pada tahapan ini, setiap istilah yang tidak menggambarkan isi dari dokumen akan dihapus, seperti kata penghubung dan kata penunjuk yang mengacu pada koleksi *stopword* pada *database*, misalnya : yang, ini, itu dan lain sebagainya. Adapun fungsi yang akan digunakan pada tahapan ini, yaitu :

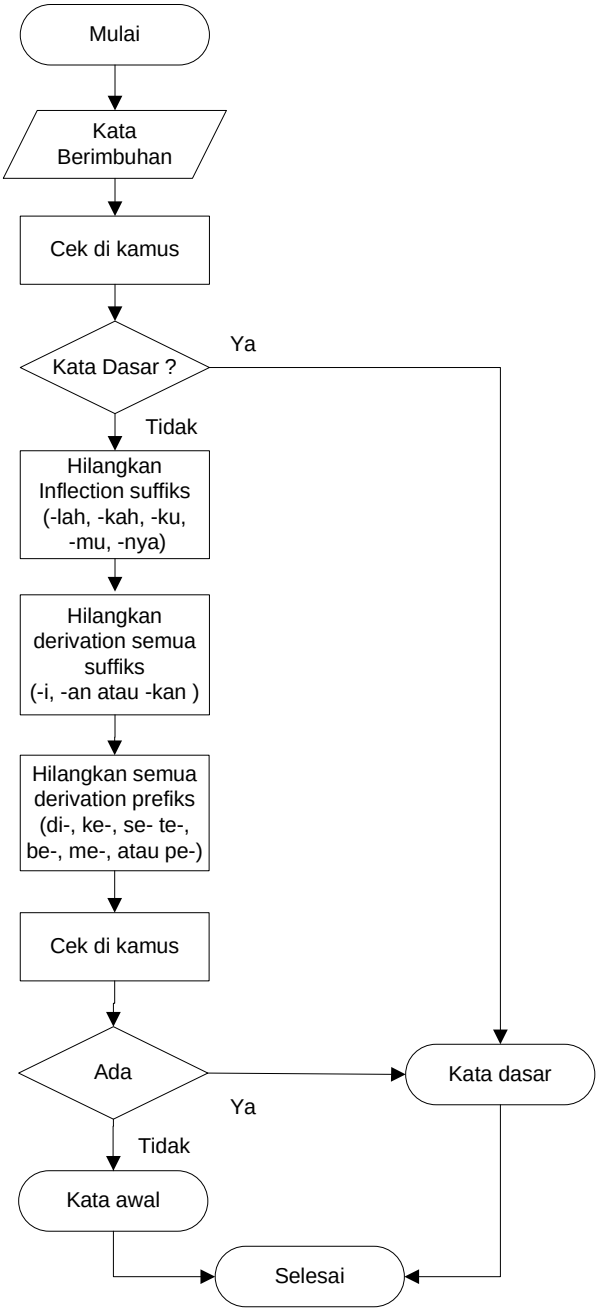
function stop_word (teks)

Masukan : dokumen dalam bentuk teks yang belum di lakukan *stop word*.

Keluaran : *stop word* pada dokumen dihapus.

e. Menerapkan *stemming* (mengembalikan kata ke kata dasar)

Dengan diterapkannya *stemming* diharapkan dapat meningkatkan performansi sistem temu kembali yang akan dibangun, hal ini dikarenakan dokumen yang akan ditemukembalikan akan lebih banyak daripada tidak menerapkan *stemming*. Adapun algoritma stemming yang akan akan digunakan yaitu algoritma Nazief & Adriani, contoh dari *stem* tersebut adalah pustaka, kembali, dan lain sebagainya. Tahapan yang dilakukan pada algortima ini dapat dijelaskan pada *flowchart* berikut.



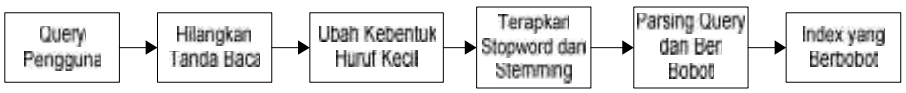
Gambar 4.2 *Flowchart* Algoritma Nazief dan Adriani

f. Pembobotan setiap istilah pada dokumen

Tahapan akhir dari preproses dokumen adalah pembobotan, dengan adanya pembobotan ini setiap kata akan di*parsing* dan dihitung jumlah

kemunculannya. Semakin besar kemunculan suatu kata dalam dokumen maka akan memberikan kesesuaian yang semakin besar.

2. Tahapan Preproses *Query* Pengguna



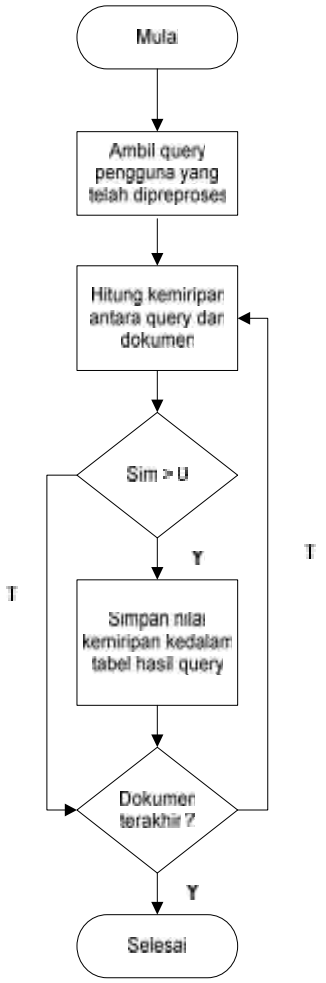
Gambar 4.3 Tahapan Preproses *Query* Pengguna

Preproses yang terjadi pada *query* pengguna secara garis besar sama dengan preproses yang terjadi pada dokumen, yaitu :

a. Menghilangkan tanda baca pada *query*
b. Mengubah *query* kebentuk huruf kecil
c. Menerapkan *stop word removal*
d. Menerapkan *stemming*
e. Parsing dokumen dan beri bobot setiap istilah pada *query* pengguna. Pembobotan pada *query* ini mengacu dari hasil *indexing* pada preproses dokumen.

3. Penerapan Model Ruang Vektor

Setelah pemberian bobot setiap istilah pada dokumen dan *query* pengguna, maka dilakukan tahapan berikut, yaitu :



Gambar 4.4 Gambar Penerapan Model Ruang Vektor

a. Hitung panjang vektor setiap dokumen dan *query*

   Sebelum dilakukan penghitungan relevansi dokumen dan *query*, setiap dokumen pada koleksi dan *query* pengguna akan dihitung panjang vektornya.

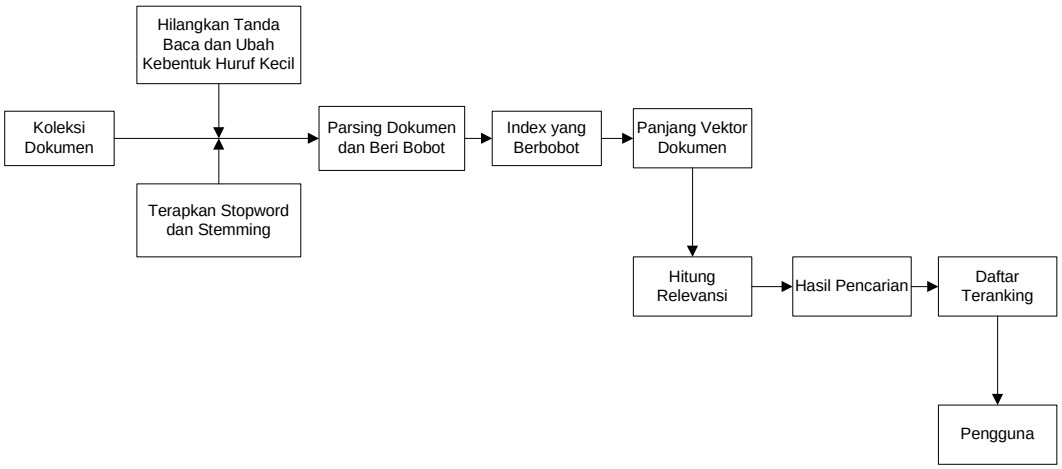
b. Hitung kedekatan (relevansi / *similirity*) antara dokumen dan *query* pengguna.

   Setelah didapatkan panjang vektor setiap dokumen dan *query* pengguna, dilakukan penghitungan kedekatan *query* tersebut terhadap dokumen yang ada pada koleksi. Dari proses ini didapatkan relevansi / *similirity* yang akan dijadikan acuan dalam menentukan dokumen yang relevan sesuai *query* yang diinputkan.

c. Simpan hasil relevansi antara dokumen dan *query* dan lakukan perangkingan.

Jika hasil dari perhitungan besar dari nol maka data perhitungan tersebut disimpan, jika tidak maka data perhitungan tidak disimpan.

Berdasarkan analisa dari tiga tahapan yang dilakukan oleh sistem *Information Retrieval*, maka dapat diilustrasikan sebagai berikut.



Gambar 4.5 Tahapan Dalam Sistem *Information Retrieval*

4.1.1 Tahapan Preproses Dokumen

Koleksi dokumen merupakan bagian utama dan pertama yang diproses sebelum dilakukan pencarian dengan menggunakan *query*, misalkan ada tiga dokumen yang dimiliki, kemudian *query* yang dimasukkan adalah rumah sakit sina. Dokumennya yaitu :

a. Dokumen d1 = Rumah Sakit Ibnu Sina.
b. Dokumen d2 = Rumah Sakit Santa Maria
c. Dokumen d3 = Rumah Sakit Bina Kasih

<u>Tahapan - tahapan yang dilakukan :</u>

1. Menghilangan tanda baca
   a. Dokumen d_1 = rumah sakit sina

2. Mengubah istilah ke bentuk huruf kecil

   a. Dokumen d_1 = rumah sakit sina

3. Menerapkan *stopword removal*

   Adapun daftar *stop word* dari tiga contoh dokumen diatas adalah : yang, dalam, dapat, untuk, tersebut, dengan, adalah, dan, bagi, besar, secara.

   a. Dokumen d_1 = rumah sakit sina

4. Menerapkan *stemming* (mengembalikan kata ke kata dasar)

   Adapun daftar *stemming* dari tiga contoh dokumen diatas adalah : milik, hasil, manfaat, selesai, bantu, dukung, putus, kumpul, basis, proses, beri, timbang, ambil, rupa, masalah, mungkin.

   a. Dokumen d_1 = rumah sakit sina

5. Pembobotan, setelah semua dokumen di*preprocessing* tiap *term* dipisah dan dimasukkan ke dalam tabel indexing.

   Dalam koleksi ini, terdapat tiga dokumen, sehingga diperoleh N = 3 dan berdasarkan rumus 2.9 maka untuk istilah algoritma dimana istilah algoritma tersebut muncul pada pada 2 dokumen yaitu pada dokumen d_1 dan d_3 maka diperoleh df = 1, idf yang didapatkan adalah 0.817249

   Pembobotan untuk istilah algoritma dapat menggunakan rumus 2.8 sehingga untuk istilah algoritma diperoleh w (bobot) = 0.817249. Dengan penerapan rumus yang sama idf dan bobot setiap istilah selengkapnya dapat dilihat pada tabel 4.1.

Tabel 4.1 Hasil pembobotan *index* dokumen

No	Kata	Id Dok	Jumlah (tf)	Idf	W = (tf.idf)
1	rumah	3	1	0.817249	0.817249
2	sakit	3	1	0.817249	0.817249
3	ibnu	3	1	2,17898	2,17898
4	sina	3	1	2,17898	2,17898
5	rumah	2	1	0.817249	0.817249
6	sakit	2	1	0.817249	0.817249
7	santa	2	1	1,87795	1,87795
8	maria	2	1	1,87795	1,87795
9	rumah	62	1	0.817249	0.817249
10	sakit	62	1	0.817249	0.817249
11	bina	62	1	2,178989	2,178989
12	kasih	62	1	2,178989	2,178989

4.1.2 Tahapan Preproses *Query*

Setelah dilakukan pengindeksan terhadap koleksi dokumen, diinputkan *query* yang akan dilakukan pencocokan terhadap koleksi dokumen. Misalkan *query* yang diinputkkan adalah “Rumah Sakit Ibnu Sina ”.

Tahapan – tahapan yang dilakukan

1. Menghilangkan tanda baca
   Rumah Sakit Ibnu Sina
2. Mengubah istilah ke bentuk huruf kecil
   rumah sakit ibnu sina
3. Menerapkan *stopword removal*
   rumah sakit ibnu sina
4. Menerapkan *stemming* (mengembalikan kata ke kata dasar)
   rumah sakit sina

5. Pembobotan, *query* yang telah di preproses dan sesuai dengan istilah hasil *indexing* pada koleksi dokumen disimpan ke dalam *indexing query*.

   Pada koleksi dokumen terdapat 151 dokumen, sehingga diperoleh N= 151 dan berdasarkan rumus 2.9 maka untuk kata rumah dimana kata tersebut muncul pada pada 23 dokumen yaitu pada dokumen d_1 sampai d_{23} maka diperoleh df = 23, idf yang didapatkan adalah 0. 817249.

   Dengan penerapan rumus yang sama maka idf setiap istilah selengkapnya dapat dilihat pada tabel 4.2

   Tabel 4.2 Hasil pembobotan *index query*

No	Query	Tf(d1)	Tf(d2)	Tf(d3)	Tf(d23)	Tf(d...151)	Df	idf
1	Rumah	1	1	1	1	0	23	0,817249
2	Sakit	1	1	1	1	0	23	0,817249
3	ibnu	0	0	1	0	0	1	2,17898
4	sina	0	0	1	0	0	1	2,17898

Penerapan rumus $Idf = \log \frac{N}{df}$

dimana :

N = 151

df rumah = 23

df sakit = 23

df ibnu = 1

df sina = 1

Menghitung nilai *Idf* rumah

$$Idf = \log \frac{151}{23}$$

$$Idf = 0{,}817249111$$

Menghitung nilai *Idf* sakit

$$Idf = \log \frac{151}{23}$$

$$Idf = 0{,}817249111$$

Menghitung nilai *Idf* ibnu

$$Idf = \log \frac{151}{1}$$

$$Idf = 2{,}17898$$

Menghitung nilai *Idf* sina

$$Idf = \log \frac{151}{1}$$

$$Idf = 2{,}17898$$

4.1.3 Penerapan Model Ruang Vektor

Setelah diperoleh bobot dari setiap istilah pada dokumen dan *query*, maka dilakukan perhitungan relevansi atau *similirity* dengan menggunakan rumus 2.6 dan hasil relevansi tersebut dirangking untuk ditampilkan kembali kepada pengguna.

$$= \frac{w_{11} x w_{q1} + w_{12} x w_{q2} + w_{13} x w_{q3} + w_{14} x w_{q4}}{\overline{wd_{11}}^{2} + \overline{wd_{12}}^{2} + \overline{wd_{13}}^{2} + \overline{wd_{14}}^{2}}$$

$$x \; wq_1{}^{2} + wq_2{}^{2} + wq_3{}^{2} + wq_4{}^{2}$$

$$= \frac{0{,}817249x\,0{,}817249 + 0{,}817249x\,0{,}817249 + 2.17898x\,2.17898 + (2.17898x\,2.17898)}{\overline{0.817249^{2} + 0.817249^{2} + 2.17898^{2} + 2.17898^{2}}}$$

$$x \; 0{,}817249^{2} + 0{,}817249^{2} + 2{,}17898^{2} + 2{,}17898^{2}$$

$$= \frac{0{,}667 + 0{,}667 + 4{,}748 + (4{,}748)}{\sqrt{10{,}832\,x\,10{,}832}}$$

$$= \frac{10{,}832}{\sqrt{10{,}832\,x\,10{,}832}}$$

$$= \mathbf{1}$$

Pengurutan Hasil

Dari hasil penghitungan relevansi pada langkah sebelumnya, maka koleksi dokumen tersebut dapat diurutkan dari yang paling relevan (diurut menurun) sebagai berikut:

Query yang dimasukkan “Rumah Sakit Ibnu Sina”. Koleksi dokumen yang ditampilkan adalah *Similarity* = 1 .

Penerapan di *google maps*

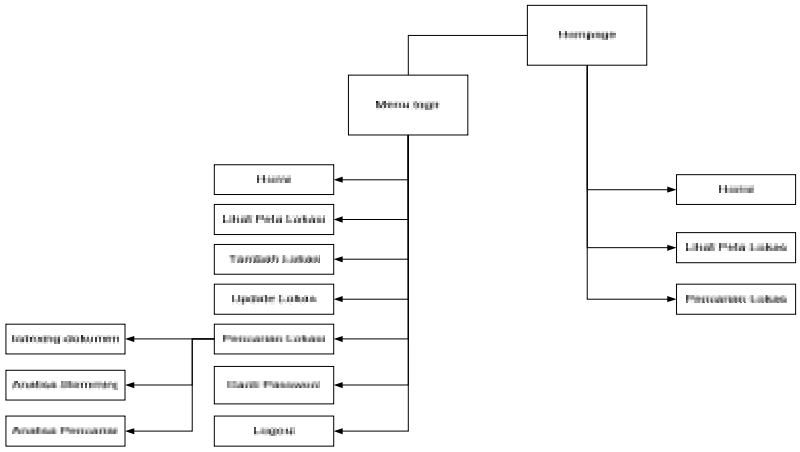
Setelah di dapatkan hasil pencarian model ruang vektor, penerapan akan dilakukan pada sistem. Yang mana sistem akan menampilkan perangkingan dokumen berdasarkan *query* inputan yang telah di proses. Dan dapat dilihat lokasi fasilitas pada *google maps.*

4.2 Perancangan Tampilan Sistem

Perancangan tampilan sistem ini dibuat dengan tujuan sebagai acuan dari tampilan implementasi dari sistem yang akan dibangun, dilain sisi perancangan sistem ini juga berguna dalam memberikan kemudahan dan kenyamanan bagi pengguna dalam mengoperasikan sistem. Adapun beberapa dari rancangan tampilan tersebut, yaitu :

1. Tampilan Rancangan Menu Sistem

   Gambar 4.6 merupakan rancangan tampilan struktur menu sistem, pengguna dapat memilih salah satu menu yang disediakan dalam pengoperasian sistem.



Gambar 4.6 Rancangan Tampilan Menu Sistem

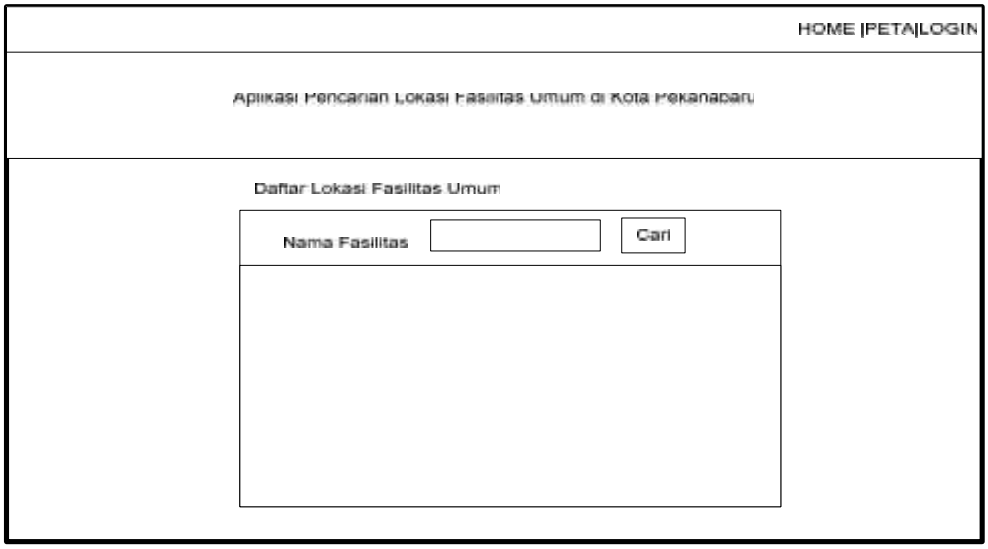
2. Rancangan Tampilan *Home User*

   Gambar 4.7 merupakan rancangan tampilan *home user*

Gambar 4.7 Rancangan Form Koleksi Dokumen

3. Tampilan Rancangan Halaman Peta *User*

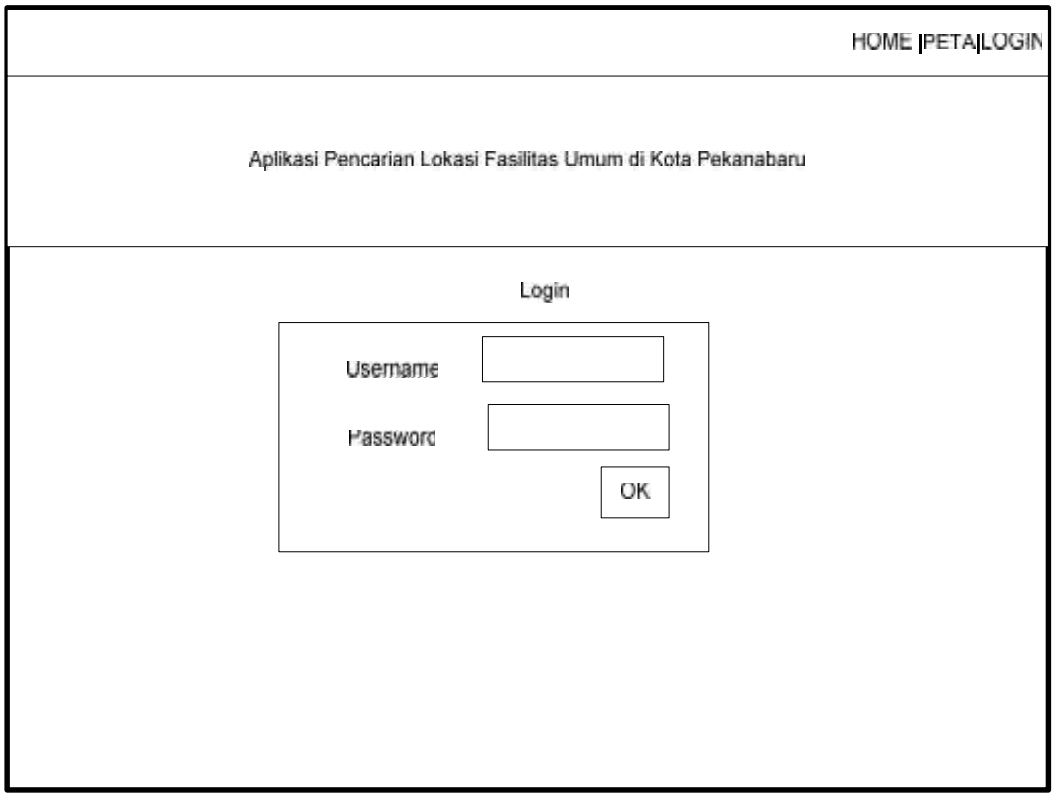
   Gambar 4.8 merupakan tampilan peta, pada halaman ini *user* dapat melahat peta kota Pekanbaru dan melakukan *zoom in* dan *zoom out* terhadapat peta. *User* juga dapat melakukan pencarian lokasi.



Gambar 4.8 Rancangan Halaman Peta *User*

4. Halaman *Login*

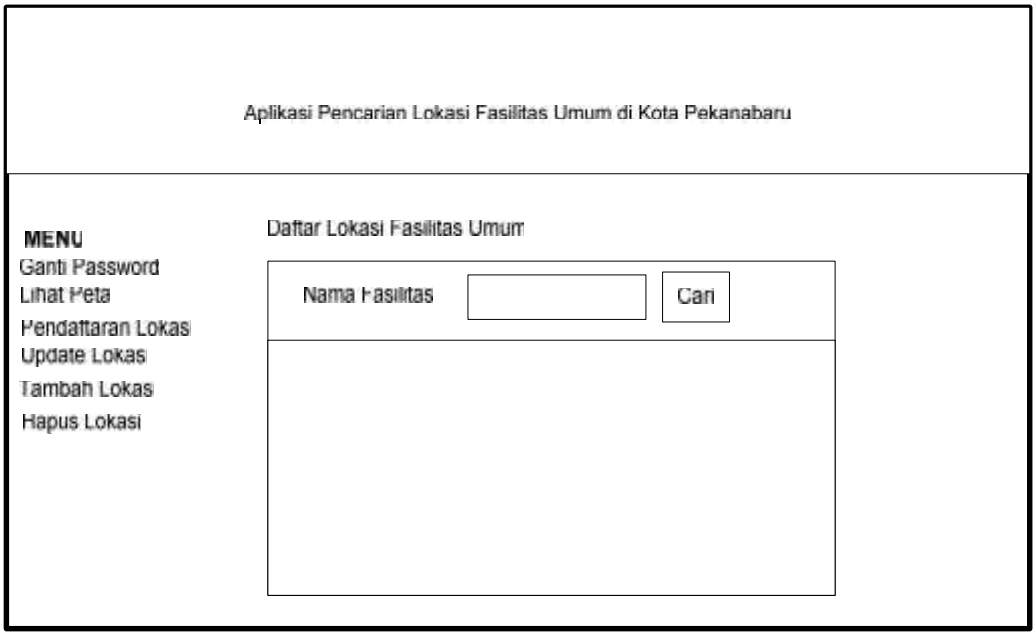
Gambar 4.9 merupakan rancangan halaman *login*. Halaman ini diperuntukkan bagi admin pengelola aplikasi. Dengan mengakses halaman *login*, admin dapat mengelola peta serta data fasilitas umum yang akan disajikan bagi masyarakat.



Gambar 4.9 Rancangan Halaman *Login Admin*

5. Tampilan Rancangan Halaman Lihat Lokasi Peta Admin

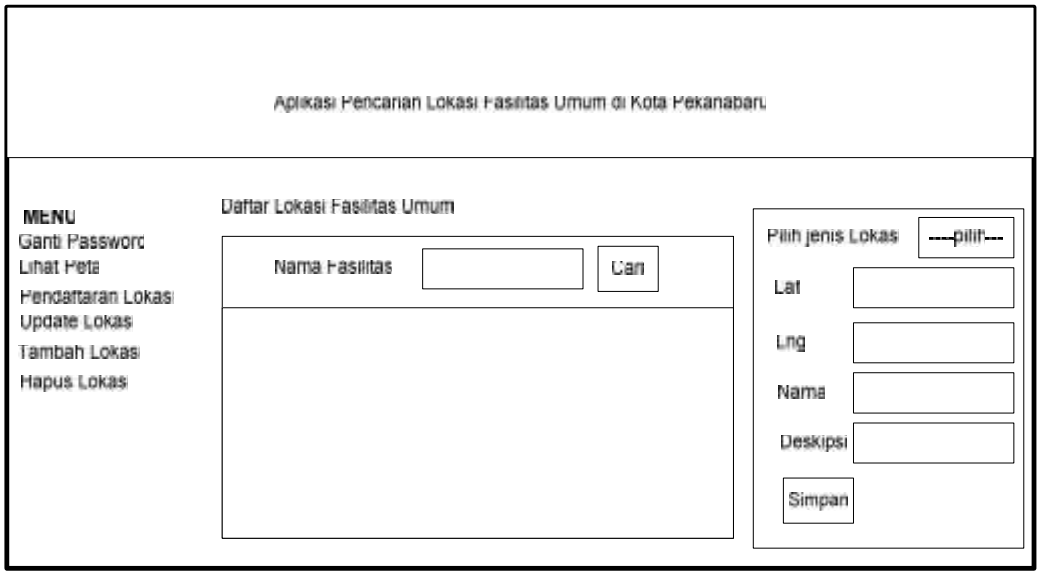
Gambar 4.10 merupakan rancangan tampilan lihat peta pada akses admin. Pada halaman ini admin dapat melihat lokasi fasilitas umum di kota Pekanbaru, beserta melakukan *zoom in* dan *zoom out* peta



Gambar 4.10 Rancangan Halaman Lihat Lokasi Peta Admin

6. Tampilan Rancangan Halaman Pendaftaran Lokasi Peta

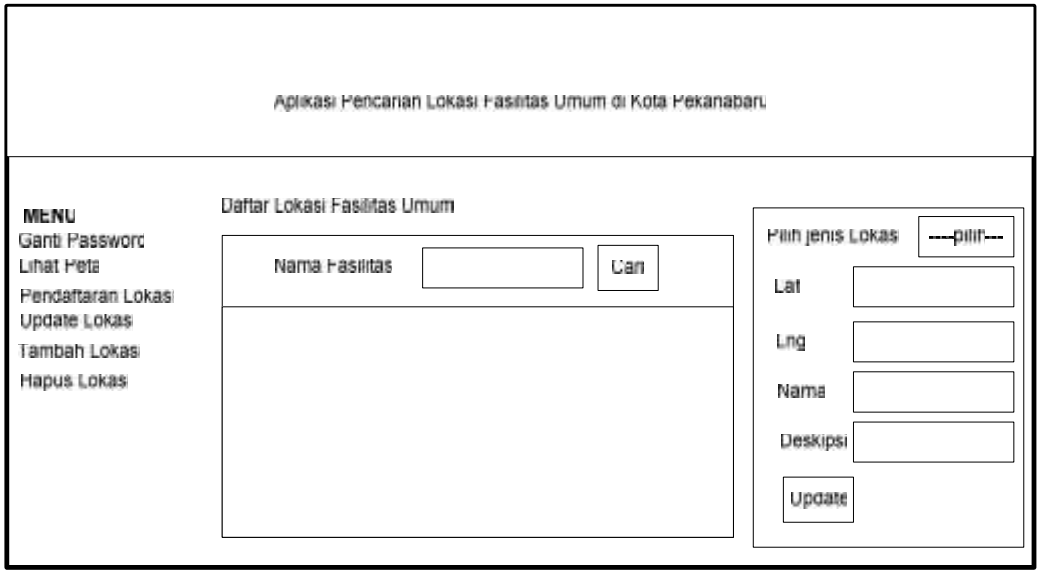
Gambar 4.10 merupakan rancangan tampilan untuk pendaftaran lokasi. Pada halaman ini admin dapat melihat lokasi fasilitas umum di kota Pekanbaru, beserta melakukan *zoom in* dan *zoom out* peta.



Gambar 4.11 Rancangan Halaman Pendaftaran Lokasi Peta

7. Tampilan Rancangan Halaman *Update* Lokasi Peta

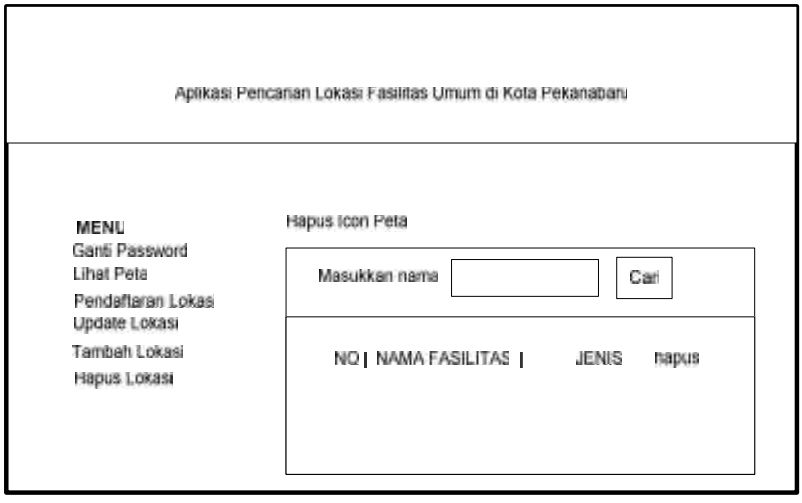
Gambar 4.10 merupakan rancangan tampilan untuk *update* lokasi. Pada halaman ini admin dapat melakukan perubahan data lokasi fasilitas umum di kota Pekanbaru, beserta melakukan *zoom in* dan *zoom out* peta.



Gambar 4.12 Rancangan Halaman *Update* Lokasi Peta

8. Tampilan Rancangan Halaman *Update* Lokasi Peta

Gambar 4.10 merupakan rancangan tampilan untuk *update* lokasi. Pada halaman ini admin dapat melakukan perubahan data lokasi fasilitas umum di kota Pekanbaru, beserta melakukan *zoom in* dan *zoom out* peta.



Gambar 4.13 Rancangan Halaman Hapus Lokasi Peta

BAB V

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dijelaskan implementasi dan pengujian dari sistem *information retrieval* yang telah dirancang pada bab sebelumnya. Adapun pembahasan pada tahapan implementasi meliputi batasan implementasi, lingkungan operasional dan hasil implementasi dan pada pengujian meliputi lingkungan pengujian, tahapan pengujian dan hasil pengujian.

5.1 Implementasi

Tahapan implementasi adalah kondisi dimana sistem yang telah analisa dan dirancang siap dioperasikan pada kondisi yang sebenarnya, dari tahapan implementasi ini akan diketahui tingkat keberhasilan analisa dan perancangan pada sistem yang akan dibangun.

5.1.1 Batasan Implementasi

Sistem information *retrieval* yang dibangun pada tugas akhir memiliki batasan sebagai berikut :

1. Bahasa pemrograman yang digunakan dalam pengimplementasian sistem ini yaitu *PHP* dengan *DBMS mySQL* pada sistem operasi *Microsoft Windows 7 Home Premium*.
2. Adapun algoritma yang digunakan dalam pengembalian istilah kekata dasar pada dokumen adalah Algoritma Nazief & Adriani dengan kamus kata dasar bahasa Indonesia.
3. Sistem *information retrieval* yang dibangun hanya menampilkan pencarian sesuai dengan *query* yang diinputkan pengguna, tidak ada penambahan kata bantu dan perluasan *query*.
4. Dalam penambahan koleksi dokumen, file yang dapat diinputkan hanya *file* dengan ekstensi pdf dan hanya teks yang dapat dikonversi dan dimasukan kedalam *DBMS mySQL*.

5.1.2 Lingkungan Operasional

Adapun pengimplementasian sistem *information retrieval* ini dibagi kedalam dua komponen yaitu perangkat keras dan perangkat lunak, berikut ini adalah lingkungan operasional yang digunakan dalam pengimplementasian sistem:

1. Perangkat keras
   Processor : *Intel(R) Core(TM) i5 CPU M 430 @2.27GHz*
   Memori (RAM) : 4.00 GB
2. Perangkat Lunak
   Sistem Operasi : *Windows 7 Home Premium*
   Bahasa Pemrograman : Php
   DBMS : mySQL
   Tools Perancangan : Notepad++

5.1.3 Hasil Implementasi

Adapun hasil implentasi sistem ini dibagi menjadi dua yaitu hasil implementasi perhitungan sistem dan hasil implementasi *interface* sistem.

5.1.3.1 Hasil Implementasi Sistem

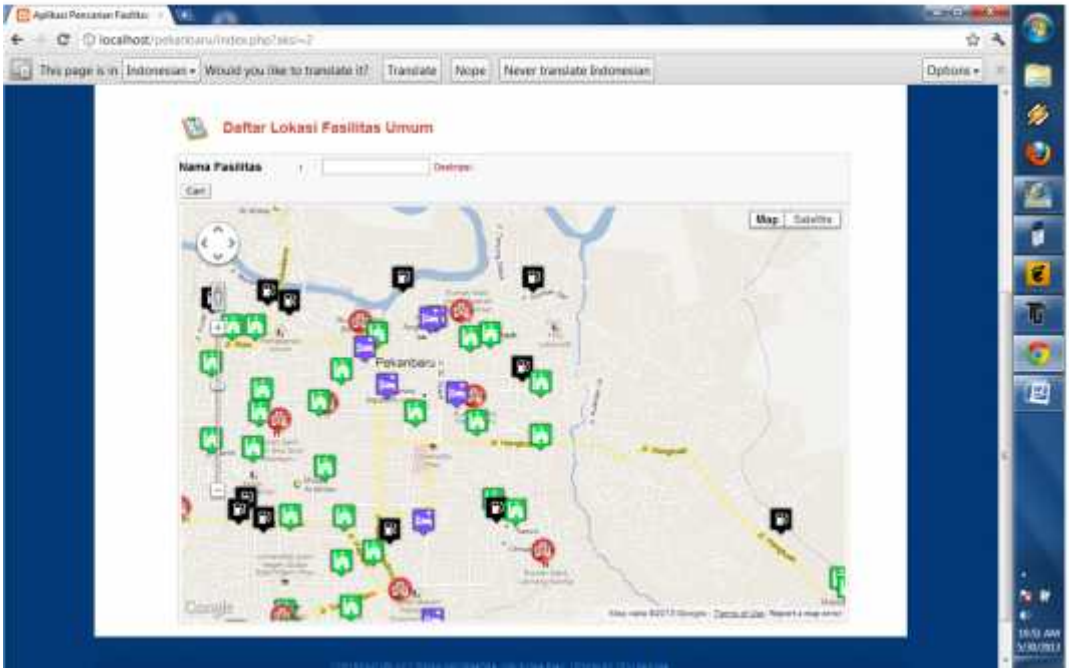
a. Halaman Utama

Gambar 5.1 adalah menu utama dari sistem yang dibangun, berawal dari menu utama inilah pengguna dapat memilih menu dan mengoperasikan sistem. Pada halaman ini teradapat 3 menu : yaitu *home,* peta, dan *login*. Pada halaman ini masyarakat dapat langsung memilih menu peta untuk melihat peta kota pekanbaru seperti pada gambar 5.1 berikut :

Gambar 5.1 Halaman Utama Aplikasi pencarian alamat fasilitas umum

b. Halaman Peta

Pada halaman peta terdapat 2 pilihan tampilan peta, yaiu tampilan map dan *satelitte* dapat dilihat pada gambar 5.2 dan gambar 5.3 berikut :

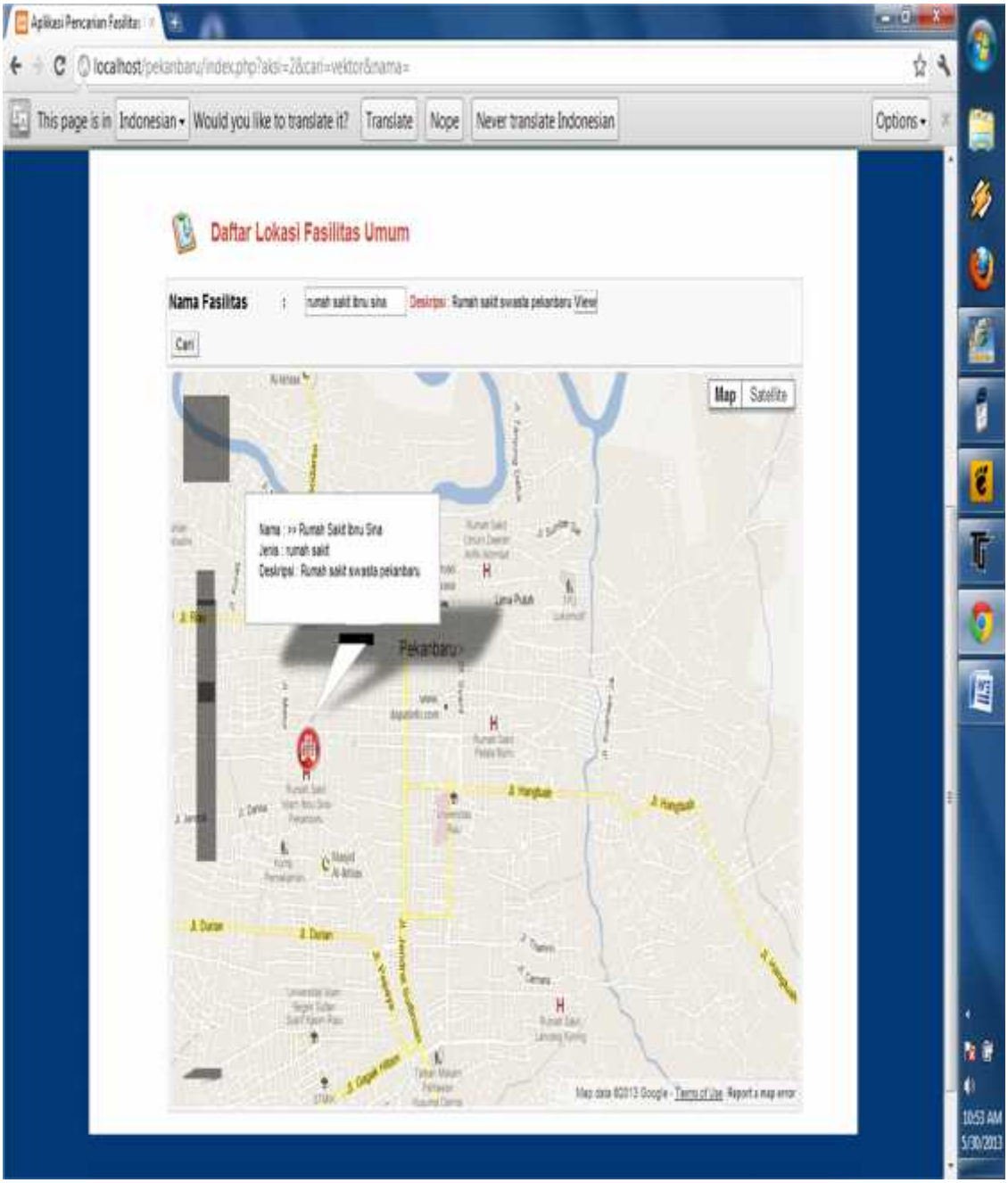


Gambar 5.2 Halaman peta mode map

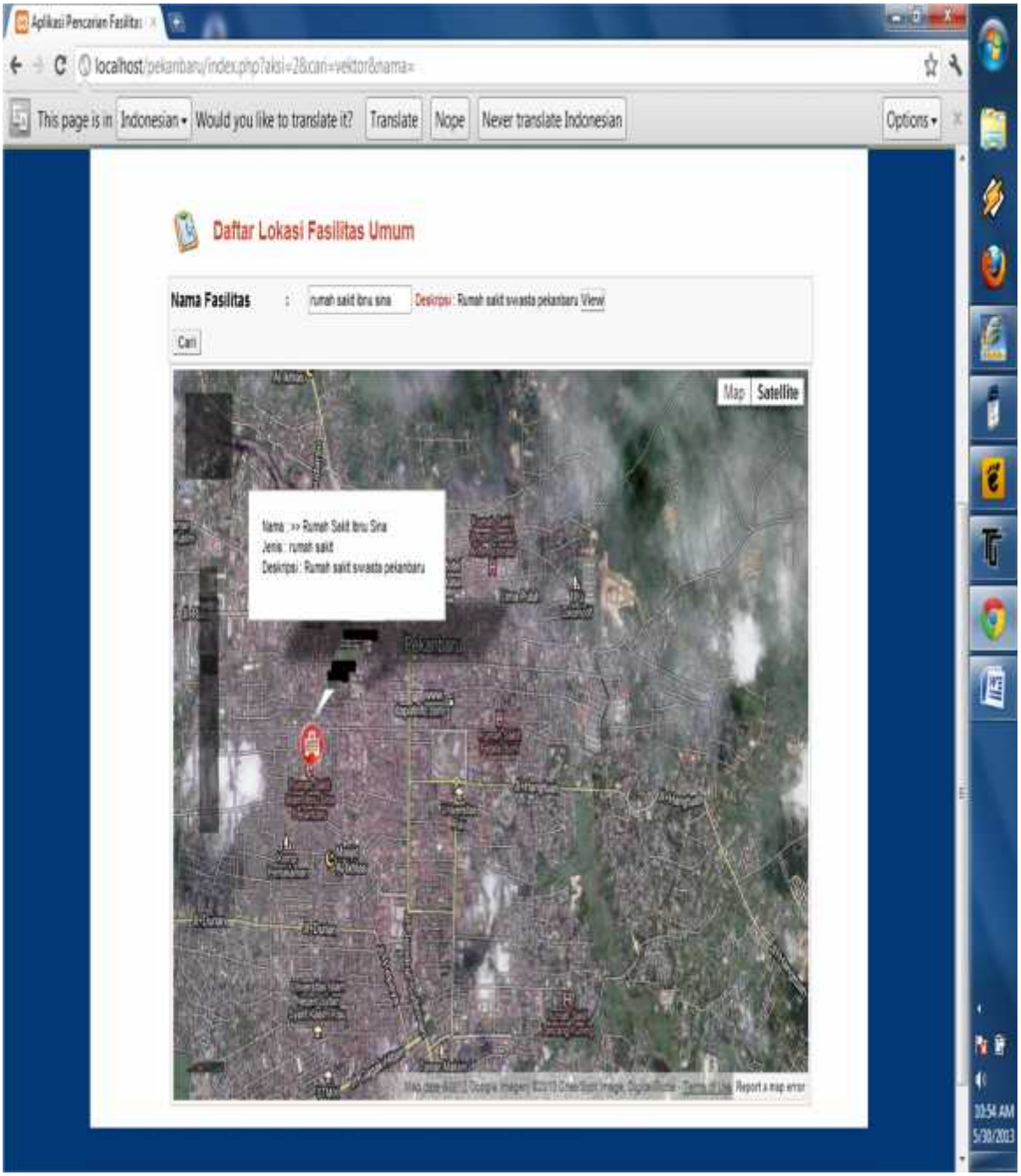
Gambar 5.2 Halaman peta mode *satellite*

c. Halaman Pecarian Lokasi

untuk halaman pencarian menggunakan mode map dan *satellite* dapat dilihat pada gambar 5.4 dan gambar 5.5 :



Gambar 5.4 Halaman pencarian lokasi mode map



Gambar 5.5 Halaman pencarian lokasi mode *satellite*

d. Halaman *Login*

5.6 Merupakan gambar halaman *login*, halaman ini diperuntukkan bagi *admin* pengelola aplikasi. Dengan mengakses halaman ini admin dapat mengelola data fasilitas umum yang ada di kota Pekanbaru seperti pada gambar 5.6 berikut :

Gambar 5.6 Halaman *Login*

e. Halaman *Home Admin*

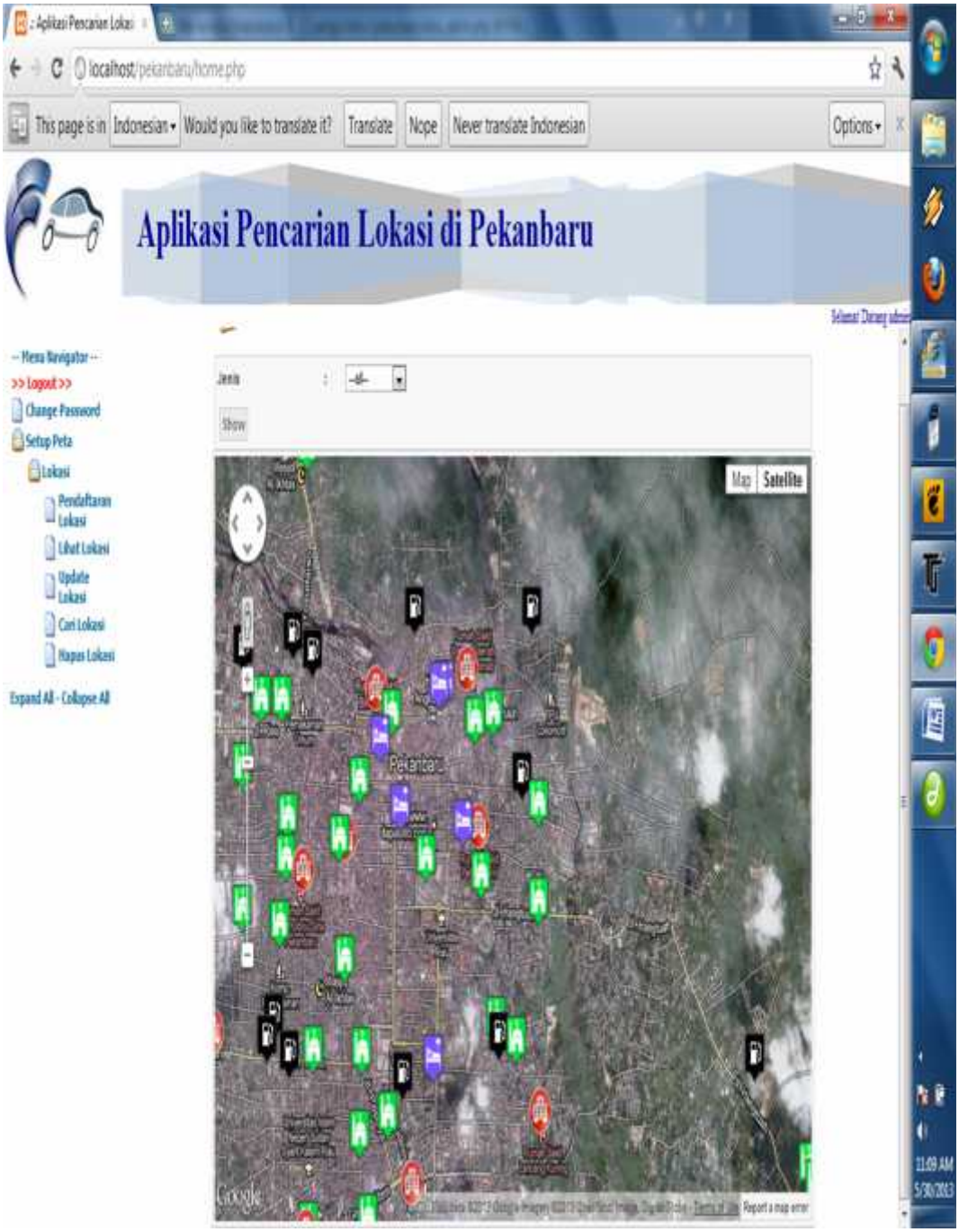
untuk halaman admin dapat dilihat seperti pada gambar 5.7 berikut :

Gambar 5.7 Halaman *Home Admin*

f. Halaman Lihat Lokasi

Untuk lihat lokasi dengan menggunakan mode map dan *satellite* dapat dilihat pada gambar 5.8 dan gambar 5.9 berikut :

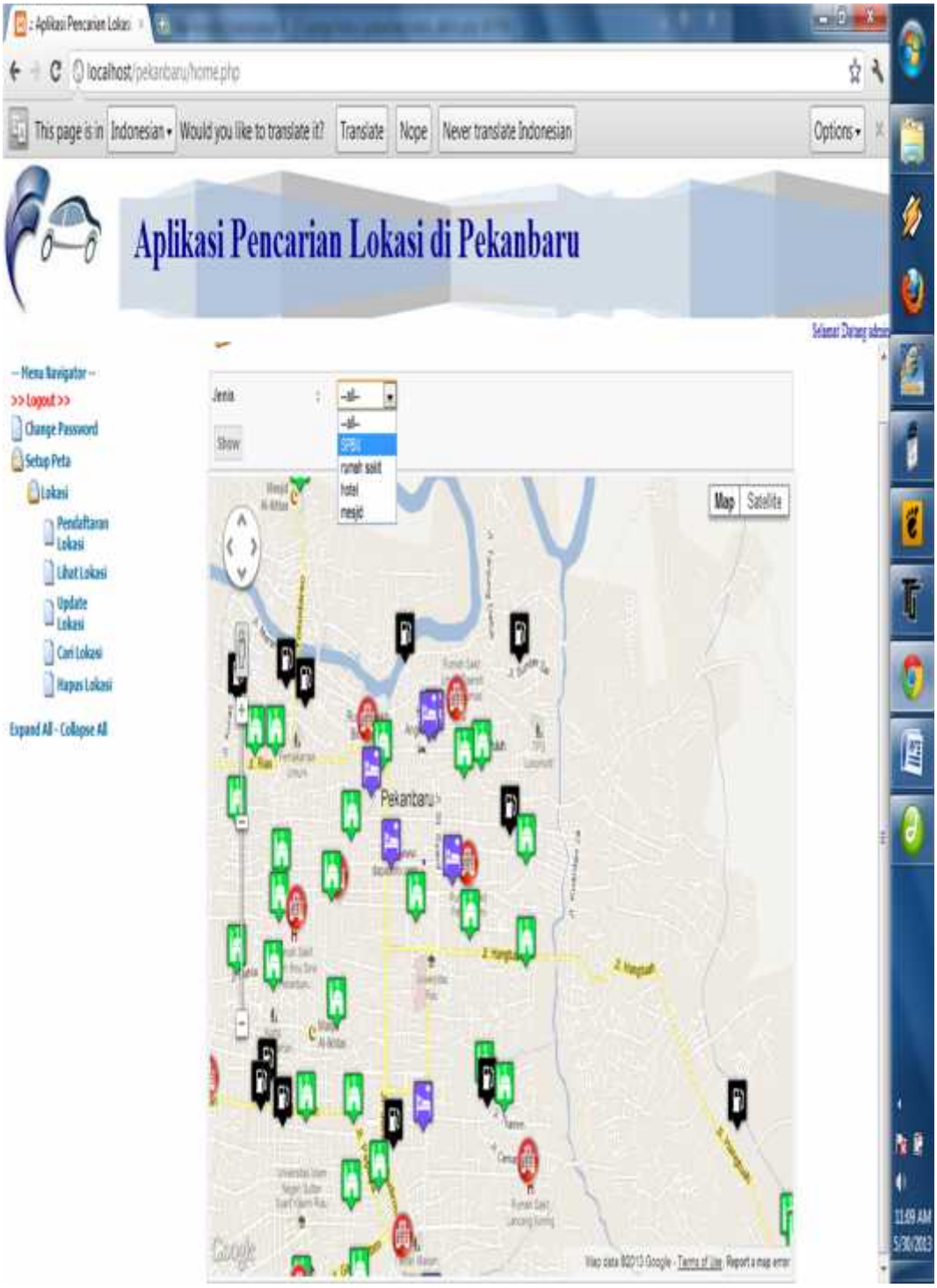
Gambar 5.8 Halaman Lihat lokasi mode map



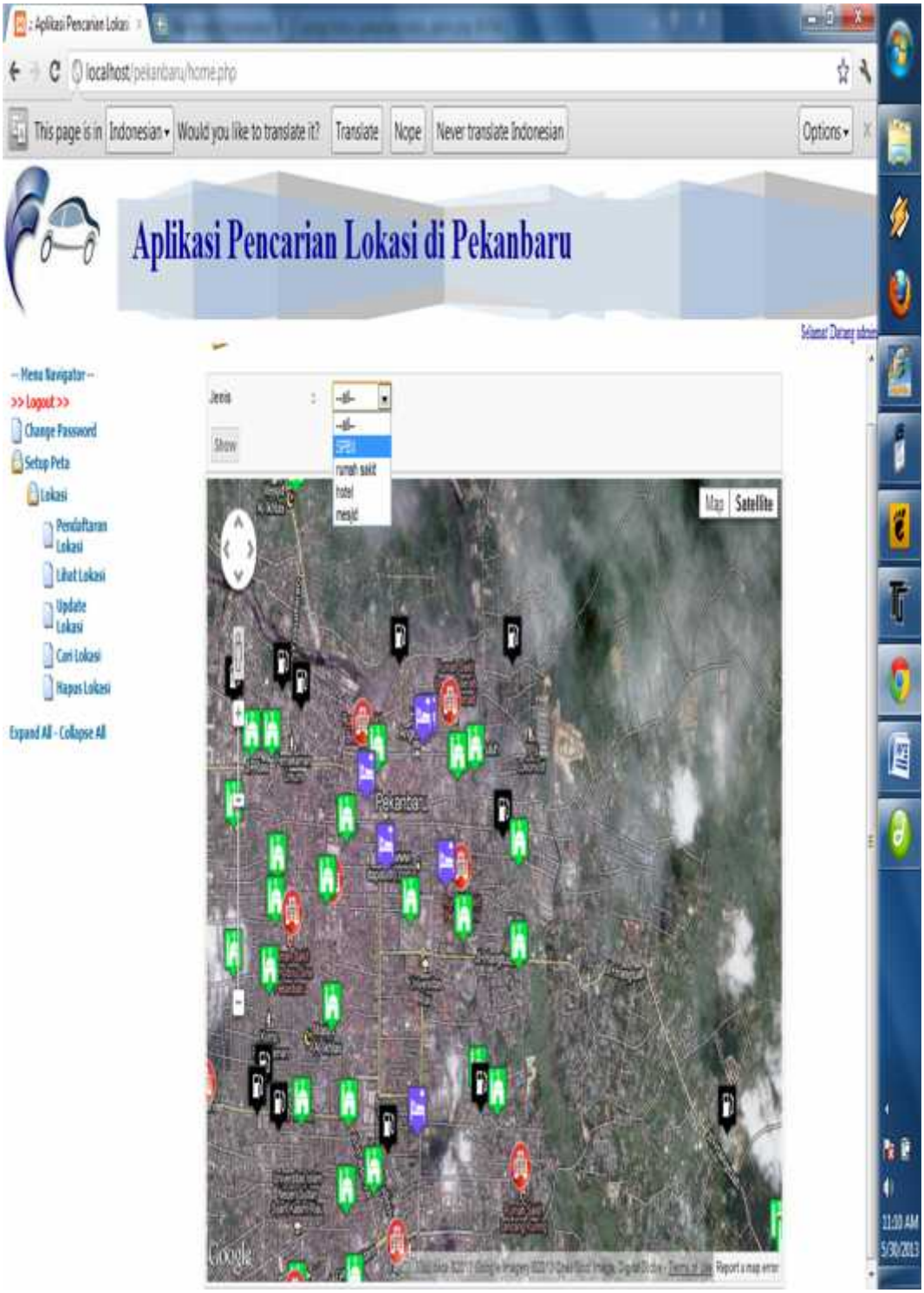
Gambar 5.9 Halaman Lihat lokasi mode *satellite*

g. Halaman Tampilan Pilihan Fasilitas

Untuk halaman tampilan pilihan fasilitas mode map dan *satellite* dapat dilihat pada gambar 5.10 dan gambar 5.11 berikut :



Gambar 5.10 Halaman tipe fasilitas mode map



Gambar 5.11 Halaman tipe fasilitas mode *satellite*

5.1.3.2 Hasil Implementasi Perhitungan Sistem

Berdasarkan perhitungan manual pada bab analisa dan perancangan maka pada bab implementasi ini akan dibandingkan kesesuaian perhitungan antara perhitungan secara manual dan perhitungan otomatis oleh sistem, misalkan terdapat tiga dokumen yang telah dilakukan *preproses* yaitu :

a. Dokumen d_1 = mesjid almubin
b. Dokumen d_2 = Mesjid Mukminin
c. Dokumen d_3 = Mesjid darul amal

Berdasarkan rumus 2.5 maka panjang vektor masing-masing dokumen dapat dilihat pada perhitungan berikut :

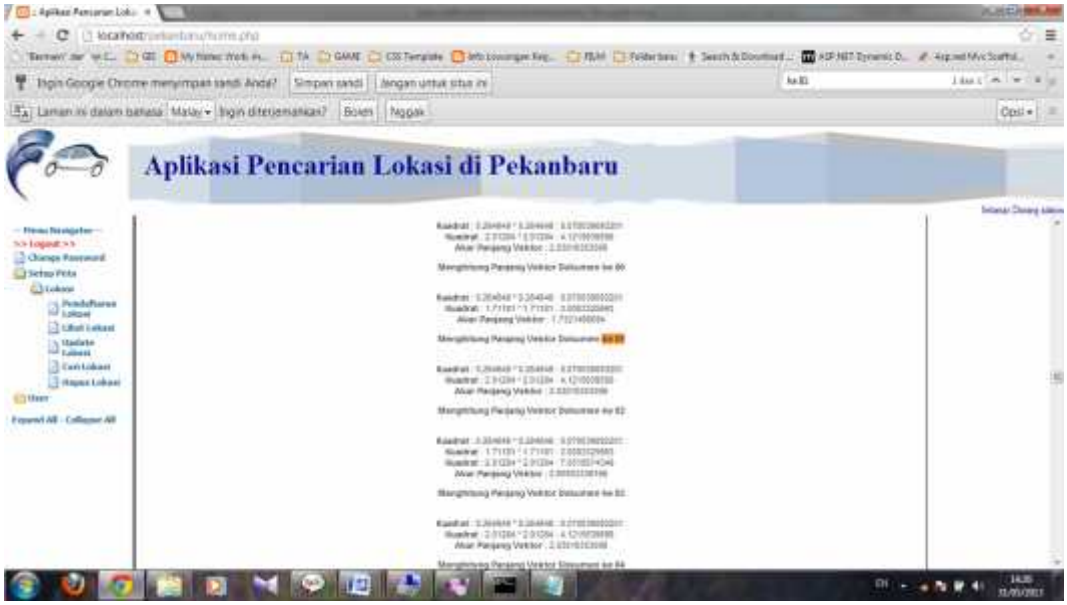
a. Menghitung Panjang Vektor Dokumen ke 81

   Kuadrat : 0.264649 * 0.264649 : 0.070039093201

   Kuadrat : 2.01284 * 2.01284 : 4.1215639588

   Akar Panjang Vektor : 2.03016353006

   Dapat dilihat pada gambar 5.12 berikut :



Gambar 5.12 Halaman akar panjang vektor *query* mesjid almubin

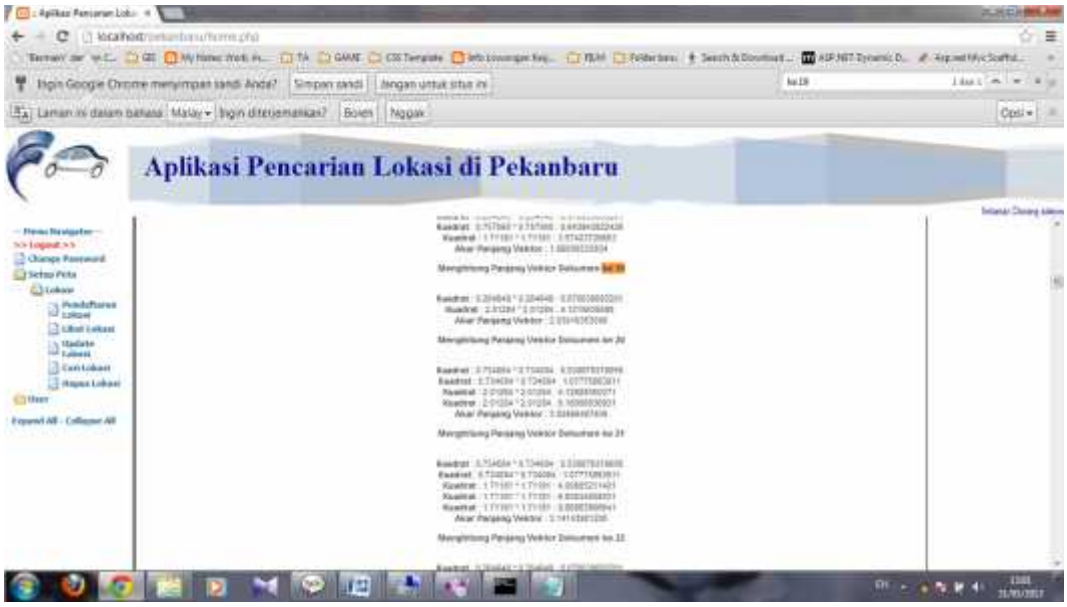
b. Menghitung Panjang Vektor Dokumen ke 19

   Kuadrat : 0.264649 * 0.264649 : 0.070039093201

   Kuadrat : 2.01284 * 2.01284 : 4.1215639588

   Akar Panjang Vektor : 2.03016353006

Dapat dilihat pada gambar 5.13 berikut :



Gambar 5.13 Halaman akar panjang vektor *query* mesjid mukminin

c. Menghitung Panjang Vektor Dokumen ke 93

Kuadrat : 0.264649 * 0.264649 : 0.070039093201

Kuadrat : 2.01284 * 2.01284 : 4.1215639588

Kuadrat : 1.71181 * 1.71181 : 7.0518574349

Akar Panjang Vektor : 2.65553336166

Dapat dilihat pada gambar 5.14 berikut :

Gambar 5.14 Halaman akar panjang vektor *query* mesjid darul amal

Adapun *query* yang diinputkan dan telah di *preproses* yaitu mesjid almubin, maka dengan menggunakan rumus 2.5 didapatkan perhitungan sebagai berikut :

Kuadrat : 0.264649 * 0.264649 : 0.070039093201

Kuadrat : 2.01284 * 2.01284 : 4.1215639588

Akar (Panjang Vektor Query) : 0.71923963075053

Tahapan terakhir adalah melakukan perhitungan relevansi menggunakan rumus 2.6 maka didapatkan perhitungan sebagai berikut :

a. Id Dok : 81

Kata : mesjid

Bobot : 0.264649

Dot Product : 0.264649 * 0.264649 : 0.070039093201

Id Dok : 81

Kata : almubin

Bobot : 2.01284

Dot Product : 2.01284 * 2.01284 : 4.1215639588

Relevansi : 4.1215639588 / (2.03016080422 * 2.03016) = 1.00000308149

Dapat dilihat pada gambar 5.15 berikut :

Gambar 5.15 Halaman analisa rumus relevansi

5.2 Pengujian

Pada bab Landasan Teori telah dijelaskan, dalam menghitung tingkat performansi sistem temu balik informasi menggunakan pengujian *recall* dan *precision*. Berdasarkan hasil dari pengujian *recall* dan *precision* tersebut, tingkat kelayakan dari sistem yang dibangun dapat diketahui.

5.2.1 Lingkungan Pengujian

Adapun pengujian sistem temu balik informasi ini dibagi kedalam dua komponen yaitu perangkat keras dan perangkat lunak, berikut ini adalah lingkungan operasional yang digunakan dalam pengujian sistem:

1. Perangkat keras

   Processor : *Intel(R) Core(TM) i5 CPU M 430 @2.27GHz*

   Memori (RAM) : 2.00 GB

2. Perangkat Lunak

   Sistem Operasi : *Windows 7 Home Premium*

   Bahasa Pemrograman : Php

   DBMS : mySQL

   Tools Perancangan : *Notepad++, Dreamweaver, Firework*

5.2.2 Rencana Pengujian

Adapun rencana pengujian yang akan dilakukan sebagai berikut :

1. Jumlah *query* yang akan dilakukan pengujian sebanyak 3 *query*.
2. Jumlah dokumen yang akan dilakukan penujian sebanyak 151 dokumen.
3. Dari pengujian *query* pada dokumen maka akan didapatkan nilai *recall* dan *precision*, dari hasil tersebut akan diketahui tingkat keberhasilan dan kelayakan sistem yang dibangun.

5.2.3 Hasil Pengujian Proses *Stemming*

Berdasarkan proses *indexing* pada sistem temu balik informasi yang telah dibangun terbentuk 428 kata, pada kata hasil *indexing* ini dilakukan pengujian *stemming* sehingga kata yang terbentuk hanya 81 kata. Jumlah kata jauh lebih

sedikit, hal ini dikarenakan pada proses pengujian *stemming* pengujian kata dilakukan secara keseluruhan. Dapat dilihat pada gambar 5.12 dan gambar 5.13 berikut :

id *	kata *	id_dok *	jumlah *	idf *	bobot *
1	rumah	1	1	0,817249	0,817249
2	sakit	1	1	0,817249	0,817249
3	petala	1	1	1,87795	1,87795
4	bumi	1	1	1,70186	1,70186
5	rumah	2	1	0,817249	0,817249
6	sakit	2	1	0,817249	0,817249
7	santa	2	1	1,87795	1,87795
8	maria	2	1	1,87795	1,87795
9	rumah	3	1	0,817249	0,817249
10	sakit	3	1	0,817249	0,817249
11	ibnu	3	1	2,17898	2,17898
12	sina	3	1	2,17898	2,17898
13	rumah	4	1	0,817249	0,817249
14	sakit	4	1	0,817249	0,817249
15	ibu	4	1	2,17898	2,17898
16	anak	4	1	1,87795	1,87795
17	labuh	4	1	2,17898	2,17898
18	rumah	5	1	0,817249	0,817249
19	sakit	5	1	0,817249	0,817249
20	test	5	1	1,87795	1,87795
21	rumah	6	1	0,817249	0,817249
22	sakit	6	1	0,817249	0,817249
23	prof	6	1	1,87795	1,87795

Gambar 5.12 Kata yang terbentuk saat *indexing*

id *	token_asli *	stem *	token_stem *	ket *
1	rumah	rumah	rumah	sukses
2	sakit	sakit	sakit	sukses
3	petala	petala	petala	sukses
4	bumi	bumi	bumi	sukses
5	santa	santa	santa	sukses
6	maria	maria	maria	sukses
7	ibnu	ibnu	ibnu	sukses
8	sina	-	sina	gagal
9	ibu	ibu	ibu	sukses
10	anak	anak	anak	sukses
11	labuh	labuh	labuh	sukses
12	test	-	test	gagal
13	prof	prof	prof	sukses
14	dr	-	dr	gagal
15	tabrani	-	tabrani	gagal
16	awal	awal	awal	sukses
17	bros	bros	bros	sukses
18	2	-	2	gagal
19	jiwa	jiwa	jiwa	sukses
20	tampan	tampan	tampan	sukses
21	iu	-	iu	gagal
22	sansani	-	sansani	gagal
23	eka	-	eka	gagal

Gambar 5.13 Kata yang terbentuk saat pengujian *stemming*

Berdasarkan Gambar 5.13, pengujian *stemming* ini memiliki tahapan sebagai berikut :

1. Lakukan *parsing* pada koleksi dokumen, kata yang di *parsing* sebelumnya telah dilakukan preproses *stop word*.
2. Dari hasil *parsing* tersebut disimpan kata asli (*token* asli), kata dasar (s*tem*), kata setelah di *stemming* (*token stem*) dan keterangan sukses atau tidak dari proses *stemming*.
3. Hasil parsing tersebut disimpan didalam tabel analisa *stem*, kata yang gagal di *stemming* dilakukan analisa secara manual, hal ini dilakukan untuk mengetahui penyebab gagalnya proses *stemming*.

4. Setelah dilakukan analisa secara manual diketahui bahwa sebagian besar kegagalan proses *stemming* dipengaruhi oleh :

   a. Istilah asing

      Adapun beberapa istilah asing yang tidak dapat dilakukan *stemming* sebagai berikut pada gambar 5.14 dan selengkapnya dapat dilihat pada **Lampiran C** :

id *	token_asli *	stem *	token_stem *	ket *
105	maghfirah	-	maghfirah	gagal
106	at	-	at	gagal
107	abdullah	-	abdul	gagal
108	faqih	-	faqih	gagal
110	nurul	-	nurul	gagal
111	ponsel	-	ponsel	gagal
112	istiqamah	-	istiqamah	gagal
113	senapelan	-	senapelan	gagal
114	istiqarah	-	istiqarah	gagal
119	annur	-	annur	gagal
120	resti	-	resti	gagal
123	grand	-	grand	gagal
125	muttaqin	-	muttaqin	gagal
127	zuri	-	zuri	gagal
128	pekabaru	-	pekabaru	gagal
129	hollywood	-	hollywood	gagal

Gambar 5.14 Istilah Asing yang tidak dapat di *stemming*

   b. Nomor

      Adapun beberapa penggunaan nomor yang tidak dapat dilakukan *stemming* pada gambar 5.15 sebagai berikut :

id *	token_asli *	stem *	token_stem *	ket *
18	2	-	2	gagal
21	iu	-	iu	gagal

Gambar 5.15 Penggunaan nomor yang tidak dapat di *stemming*

5. Tahapan selanjutnya adalah mengevaluasi dan memastikan keberhasilan dari kata-kata yang sukses di *stemming*.

   Adapun beberapa hasil *stemming* yang sukses dapat dilihat pada gambar 5.16 berikut :

Jumlah Stem Sukses (Keseluruhan) : 81 kata

Jumlah Stem Sukses : 81 kata

No	Kata Awal	Kata Dasar	Stem	Ket
1	rumah	rumah	rumah	sukses
2	sakit	sakit	sakit	sukses
3	petala	petala	petala	sukses
4	bumi	bumi	bumi	sukses
5	santa	santa	santa	sukses
6	maria	maria	maria	sukses
7	ibnu	ibnu	ibnu	sukses
8	ibu	ibu	ibu	sukses
9	anak	anak	anak	sukses
10	labuh	labuh	labuh	sukses
11	prof	prof	prof	sukses
12	awal	awal	awal	sukses
13	bros	bros	bros	sukses

Gambar 5.16 Kata-kata yang berhasil di *stemming*

Dari hasil evaluasi didapat kesimpulan presentasi kesuksesan dari penggunaan algoritma Nazief dan Adriani yaitu :

$$\text{Presentasi Sukses} = \frac{\text{Jumlah kata yang berhasil di } stemming}{\text{Jumlah total kata}}$$

$$= \frac{81}{89}\ x\ 100\% = 91{,}01\ \%$$

6. Menganalisa kembali kata-kata yang gagal di *stemming*, adapun analisanya dapat dijelaskan sebagai berikut :
    a. Algoritma Nazief dan Adriani melakukan proses *stemming* dengan menghilangkan akhiran terlebih dahulu, kemudian diikuti dengan penghilangan awalan. Langkah ini dapat menghasilkan hasil *stemming* yang tidak tepat pada beberapa kata, misalnya pada kata “sejumlah” dan “melangkah”. Hasil stemming algoritma Nazief dan Adriani akan menghasilkan kata “sejum” dan “lang”. Padahal kata dasar yang tepat dari kata “sejumlah” dan “melangkah” adalah “jumlah” dan “langkah”. Hal ini dikarenakan algoritma bekerja dengan melakukan penghilangan akhiran terlebih dahulu.
    b. Algoritma Nazief dan Adriani melakukan pengecekan awalan kata salah satunya hanya terbatas pada awalan te- dan pe- sehingga untuk kata-kata yang memiliki awalan ter- seperti “tertentu” dan pe- dengan kombinasi -ng seperti “pengenalan” tidak dapat dilakukan *stemming*.
    c. Algoritma Nazief dan Adriani juga melakukan pengecekan dengan awalan me- dengan kombinasi -m-. Namun, jika diinputkan kata “memilih”, dimana me- adalah awalan dan –m- adalah kombinasinya, algoritma Nazief dan Adriani belum mampu mengembalikan “memilih” ke kata dasarnya menjadi “pilih”.
    d. Untuk kata-kata ambigu seperti “beruang” dapat diatasi dengan menambah koleksi kata dasar “beruang”, sehingga kata dasar “beruang” dan “uang” dapat dipisahkan.

    Adapun beberapa hasil *stemming* yang gagal dapat dilihat pada gambar 5.17 berikut dan selengkapnya dapat dilihat pada **Lampiran C** :

id *	token_asli *	stem *	token_stem *	ket *
8	sina	-	sina	gagal
12	test	-	test	gagal
14	dr	-	dr	gagal
15	tabrani	-	tabrani	gagal
18	2	-	2	gagal
21	iu	-	iu	gagal
22	sansani	-	sansani	gagal
23	eka	-	eka	gagal
25	mesjid	-	mesjid	gagal
26	jamaalul	-	jamaalul	gagal
28	darul	-	darul	gagal
30	al	-	al	gagal
31	nujahidin	-	nujahidin	gagal
32	muawawanah	-	muawawanah	gagal
33	spbu	-	spbu	gagal
34	ababil	-	ababil	gagal
35	tesssss	-	tesssss	gagal
41	arrahim	-	arrahim	gagal
43	baitul	-	baitul	gagal
45	mubin	-	mubin	gagal

Gambar 5.17 Kata-kata yang gagal di *stemming*

Dari hasil evaluasi didapat kesimpulan presentasi kegagalan dari penggunaan algoritma Nazief dan Adriani yaitu :

$$\text{Presentasi Sukses} = \frac{\text{Jumlah kata yang gagal di } stemming}{\text{Jumlah total kata}}$$

$$= \frac{89}{170}\ x\ 100\% = 52{,}35\%$$

7. Menampilkan hasil analisa *stemming* kedalam sebuah grafik
   Untuk melihat hasil analisa *stemming* pada grafik dapat terlihat pada gambar 5.18 berikut :

Gambar 5.18 Grafik analisa hasil *stemming*

5.2.4 Hasil Pengujian Unjuk Kerja Sistem

Berikut daftar *query* yang diinputkan untuk dilakukan pengujian.

Tabel 5.1. Daftar *query* yang diinputkan

No	***Query***	**Jumlah dokumen relevan dalam *corpus***
Q_1	duta arya	1
Q_2	sumber sari	1
Q_3	almubin	1

Setelah *query* diinputkan, sistem akan menghitung relevansi dengan koleksi dokumen, berdasarkan perhitungan rumus *recall* dan *precision (interpolated precision)* pada rumus 2.8 dan rumus 2.9 maka jika diinputkan query Q_1, jumlah dokumen yang dikembalikan adalah 23 dokumen dengan dokumen relevan 1 dokumen dan dokumen yang tidak relevan dikembalikan sebanyak 0 dokumen.

Hasil perhitungan *recall* dan *presicion* selengkapnya dapat dilihat pada tabel 5.2.

Tabel 5.2. Perbandingan hasil perhitungan *presicion(P)* dan *recall(R)pada* Q_1

Rank	***Id doc***	***Relevant?***	***Precision (P) / Recall (R)***
1	97	*Yes*	P = 1/1 = 1 ; R = 1/1 = 1

Pengujian berikutnya dilakukan pada Q_2, setelah *query* diinputkan, sistem akan menghitung relevansi dengan koleksi dokumen, berdasarkan perhitungan rumus *recall* dan *precision (interpolated precision)* pada rumus 2.8 dan rumus 2.9 maka jika diinputkan *query* Q_2, jumlah dokumen yang dikembalikan adalah 23 dokumen dengan dokumen relevan 23 dokumen dan dokumen yang tidak relevan dikembalikan sebanyak 0 dokumen.

Hasil perhitungan *recall* dan *presicion* selengkapnya dapat dilihat pada tabel 5.3.

Tabel 5.3. Perbandingan hasil perhitungan *presicion(P)* dan *recall(R) pada* Q_2

Rank	***Id doc***	***Relevant?***	***Precision (P) / Recall (R)***
1	41	*Yes*	P = 1/1 = 1 ; R = 1/1 = 1
2	55	*Yes*	P = 2/2 = 1 ; R = 2/2 = 1

Selanjutnya dilakukan pengujian pada Q_3, setelah *query* diinputkan, sistem akan menghitung relevansi dengan koleksi dokumen, berdasarkan perhitungan rumus *recall* dan *precision (interpolated precision)* pada rumus 2.8 dan rumus 2.9 maka jika diinputkan *query* Q_3, jumlah dokumen yang dikembalikan adalah 8 dokumen dengan dokumen relevan 6 dokumen dan dokumen yang tidak relevan dikembalikan sebanyak 2 dokumen.

Hasil perhitungan *recall* dan *presicion* selengkapnya dapat dilihat pada tabel 5.4.

Tabel 5.4 Perbandingan hasil perhitungan *presicion(P)* dan *recall(R)* pada Q_3

Rank	***Id doc***	***Relevant?***	***Precision (P) / Recall (R)***
1	81	*Yes*	P = 1/1 = 1 ; R = 1/1 = 1

Berikut pengujian yang tidak menggunakan *presicion* yang tidak terinterpolasi dimana pada pengujian akan terlihat jumlah dokumen yang di-*retrieve* baik yang relevan maupun tidak dan jumlah dokumen yang tidak ter-*retrieve* baik yang relevan maupun tidak, dimana data tersebut akan digunakan untuk menghitung nilai *precision* dan *recall* berdasarkan masing-masing *query*.

Tabel 5.5. Hasil pengujian *presicion(P)* dan *recall(R)* pada Q_1

	Relevant	***Nonrelevant***
Retrieved	1 (*tp*)	0 (*fp*)
Not retrieved	0 (*fn*)	0 (*tn*)

Berdasarkan Tabel 5.5, ditunjukkan bahwa jumlah dokumen yang dikembalikan yang relevan dengan *query* (*tp*) sebanyak 1 dokumen, sedangkan dokumen yang tidak relevan (*fp*) sebanyak 0 dokumen. Dan untuk jumlah dokumen yang tidak dikembalikan yang relevan dengan *query* (*fn*) sebanyak 0 dokumen, sedangkan dokumen yang tidak relevan sebanyak (*tn*) 0 dokumen. Maka, nilai *precision* dan *recall* untuk Q_1 adalah:

Precision $\rightarrow$ $P = tp / (tp + fp) = 1 / (1+0) = 1/1 = 1$

Recall $\rightarrow$ $R = tp / (tp + fn) = 1/ (1+0) = 1/1 = 1$

Tabel 5.6. Hasil pengujian *presicion(P)* dan *recall(R)* pada Q_2

	Relevant	***Nonrelevant***
Retrieved	2 (*tp*)	0 (*fp*)
Not retrieved	0 (*fn*)	0 (*tn*)

Berdasarkan Tabel 5.6, diperoleh nilai *precision* dan *recall* sebagai berikut :

Precision $\rightarrow$ $P = tp / (tp + fp) = 2 / (2+0) = 2/2 = 1$

Recall $\rightarrow$ $R = tp / (tp + fn) = 2 / (2+0) = 2/2 = 1$

Tabel 5.7. Hasil pengujian *presicion(P)* dan *recall(R)* pada Q_2

	Relevant	***Nonrelevant***
Retrieved	1 (*tp*)	0 (*fp*)
Not retrieved	0 (*fn*)	0 (*tn*)

Berdasarkan Tabel 5.7, diperoleh nilai *precision* dan *recall* sebagai berikut :

Precision $\rightarrow$ $P = tp / (tp + fp) = 1 / (1+0) = 1/1 = 1$

Recall $\rightarrow$ $R = tp / (tp + fn) = 1 / (1+0) = 1/1 = 1$

5.2.4 Kesimpulan Pengujian Unjuk Kerja Sistem

Hasil pengujian yang diperoleh dari sistem temu balik informasi yang menggunakan model ruang vektor sebagai berikut :

1. Persentase kualitas temu balik informasi yang terjadi pada Q_1 terhadap jumlah dokumen yang di temu balikkan oleh sistem yaitu *precision* 100% dan *recall* 100%.
2. Persentase kualitas temu balik informasi yang terjadi pada Q_2 terhadap jumlah dokumen yang di temu balikkan oleh sistem yaitu *precision* 100% dan *recall* 100%.
3. Persentase kualitas temu balik informasi yang terjadi pada Q_3 terhadap jumlah dokumen yang di temu balikkan oleh sistem yaitu *precision* 100% dan *recall* 100%.
4. Presentas kualitas temu balik informasi pada keseluruhan dokumen oleh sistem *precision* dan *recall* mencapai 100% karena faktor dokumen tidak ada yang sama.

BAB VI

PENUTUP

Pada bab ini akan diuraikan beberapa kesimpulan dari hasil yang didapatkan selama penelitian dan saran yang dapat digunakan pada penelitian selanjutnya.

6.1 Kesimpulan

Setelah menyelesaikan tahapan-tahapan penelitian sistem temu balik informasi, dapat diambil beberapa kesimpulan, yaitu :

1. Berdasarkan penelitian yang telah dilakukan, model ruang vektor yang digunakan memberikan hasil yang baik, selain dapat melakukan perangkingan dokumen, model ini juga mendukung pencarian dokumen yang *partial matching*.
2. Pada pencarian belum bisa menampilkan alamat di *google maps* secara langsung jika kata yang diinputkan tidak tepat, tetapi mesti memasukkan hasil pencarian kedalam inputan agar langsung menampilkan lokasi.
3. Manajemen basisdata yang digunakan adalah *MySQL*, dalam hal ini sistem yang dibangun berjalan sedikit lambat dikarenakan sistem harus menelurusi data yang ada sehingga waktu yang diperlukan cukup lama dalam menemukan dokumen yang relevan.
4. Berdasarkan penelitian yang telah dilakukan, penerapan *stemming* pada *query* dan dokumen meningkatkan hasil pencarian terhadap dokumen yang relevan.
5. Persentase kualitas temu balik informasi yang terjadi pada *query* terhadap jumlah dokumen yang di temubalikkan oleh sistem yaitu *precision* dan *recall* memungkinkan mencapai 100%, karna tidak adanya data yang mirip.
6. Faktor pendukung keakuratan dari proses *stemming* dengan algoritma Nazeif & Adriani adalah penggunaan kamus *stemming* yang lengkap dan akurat,

algoritma ini masih terdapat beberapa kelemahan, dimana masih adanya beberapa istilah yang gagal dilakukan *stemming*.

6.2 Saran

Berdasarkan penelitian yang telah dilakukan, adapun saran-saran yang dapat dilakukan untuk perbaikan dan pengembangan sistem temu balik informasi mendatang, yaitu :

1. Sistem *Information Retrieval* yang dibangun sebaiknya tidak menggunakan manajemen database *mySQL*, namun menggunakan *file* berbasis *text* sebagai tempat penyimpanan data, hal ini akan berpengaruh pada performansi kecepatan dari proses pencarian dokumen yang relevan.
2. Adanya perluasan terhadap *query* yang diinputkan pengguna, dengan adanya perluasan *query* ini diharapkan hasil terhadap dokumen yang relevan lebih banyak ditemubalikkan.
3. Menggunakan algoritma *Enhanced Confix Stripping (ECS) Stemmer* yang merupakan algoritma perbaikan dari algoritma *Confix Stripping* (CS) *Stemmer* dengan referensi dari algoritma *stemming Nazief-Adriani* dan *Arifin-Setiono*.

DAFTAR PUSTAKA

Agusta, Lady. 2009. "*Perbandingan Algoritma Stemming Porter Dengan Algoritma Nazief & Adriani Untuk Stemming Dokumen Teks Bahasa Indonesia*". Konferensi Nasional Sistem dan Informatika 2009, Bali, November 14, 2009.

Benisius. 2011. *Sistem Pengkoreksian Kata Kunci Dengan Menggunakan Metode Levenshtein Distance.* Universitas Halmahera.

Frakes, William B., Riardo Baeza-Yates. 1992. *Information Retrieval Data Structures & Algorithms*. Prentice Hall.

Grossman, David A., Ophir Frieder. 1998. *Information Retrieval : Algorithms and Heuristics*. Kluwer Academic Publisher.

Husni., 2010, "*Sistem Temu-Balik Informasi*", diktat kuliah, Teknik Informatika Universitas Trunojoyo.

Jaya, Hendra., 2007. "*Perbandingan Performansi Word Indexing dan Phrase Indexing dalam Sistem Temu Balik Informasi dengan Menggunakan Model Probabilistik."* Skripsi Terpublikasi. Bandung : Institut Teknologi Bandung.

Mandala, Rila. 2003. Slide Kuliah Sistem Temu Balik Informasi : *Model Ruang Vektor*.

Ramadhany., Taufik. 2008. "*Implementasi Kombinasi Model Ruang Vektor dan Model Probabilistik Pada Sistem Temu Balik Informasi."* Skripsi Terpublikasi. Bandung : Institut Teknologi Bandung.

Wijarnako, Anthonius Koko., 2004. *Penerapan Metode Tanya Jawab pada Sistem Temu Balik Informasi*. Institut Teknologi Bandung.